Trie: дерево, но не такое

- 2001 — 2013: Java @ МФТИ
- 2004 — 2010: Java @ NetCracker
- 2011 — 2011: Java @ Яндекс
- 2011 — 2014: Java @ Одноклассники
- 2014 — 2015: Java @ Republer
- с 2015: Java @ СберТех

Trie

- Что такое Trie
- Как "посчитать" авторов Википедии с помощью Trie
- Как проверить соответствие строки 287 тыс. шаблонов за 1 миллисекунду

Структуры данных в Java

- Можно реализовать любые

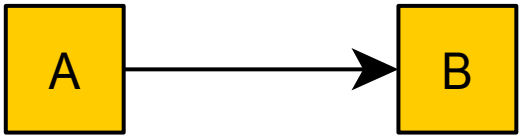
Структуры данных в Java

- Можно реализовать любые
- Уже есть в JDK:
  - *List*, *Queue*, *Deque*, *RandomAccess*, *SortedList*
  - *Set*, *SortedSet*, *NavigableSet*
  - *Map*, *SortedMap*, *NavigableMap*

Структуры данных в Java

- Можно реализовать любые
- Уже есть в JDK:
  - ArrayList, LinkedList, ...
  - HashSet, LinkedHashSet, TreeSet, ConcurrentSkipListSet, CopyOnWriteArraySet, ...
  - HashMap, TreeMap, LinkedHashMap, ConcurrentHashMap, ConcurrentSkipListMap, ...

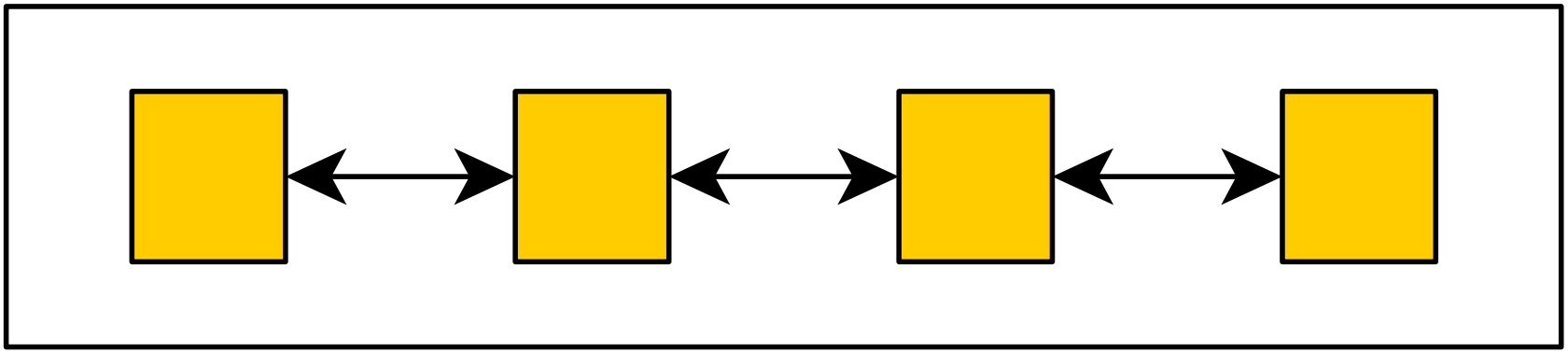
Структуры данных в Java



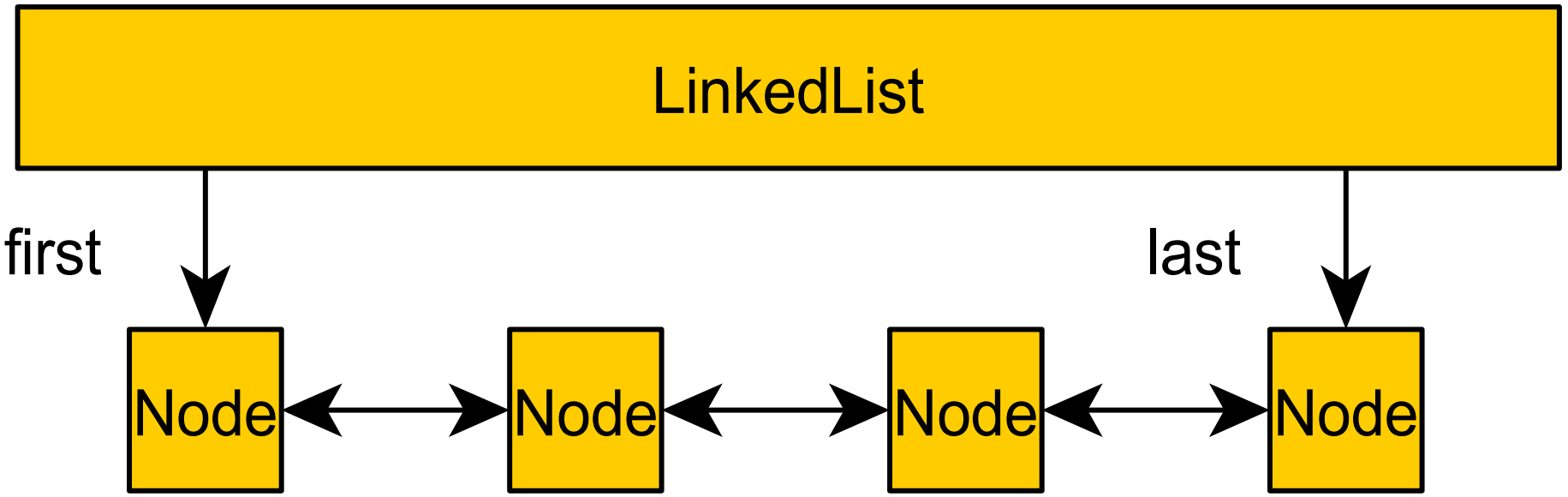

```
public class A {

    private B b;

    …

}
```

```
public class B {

    …

}
```

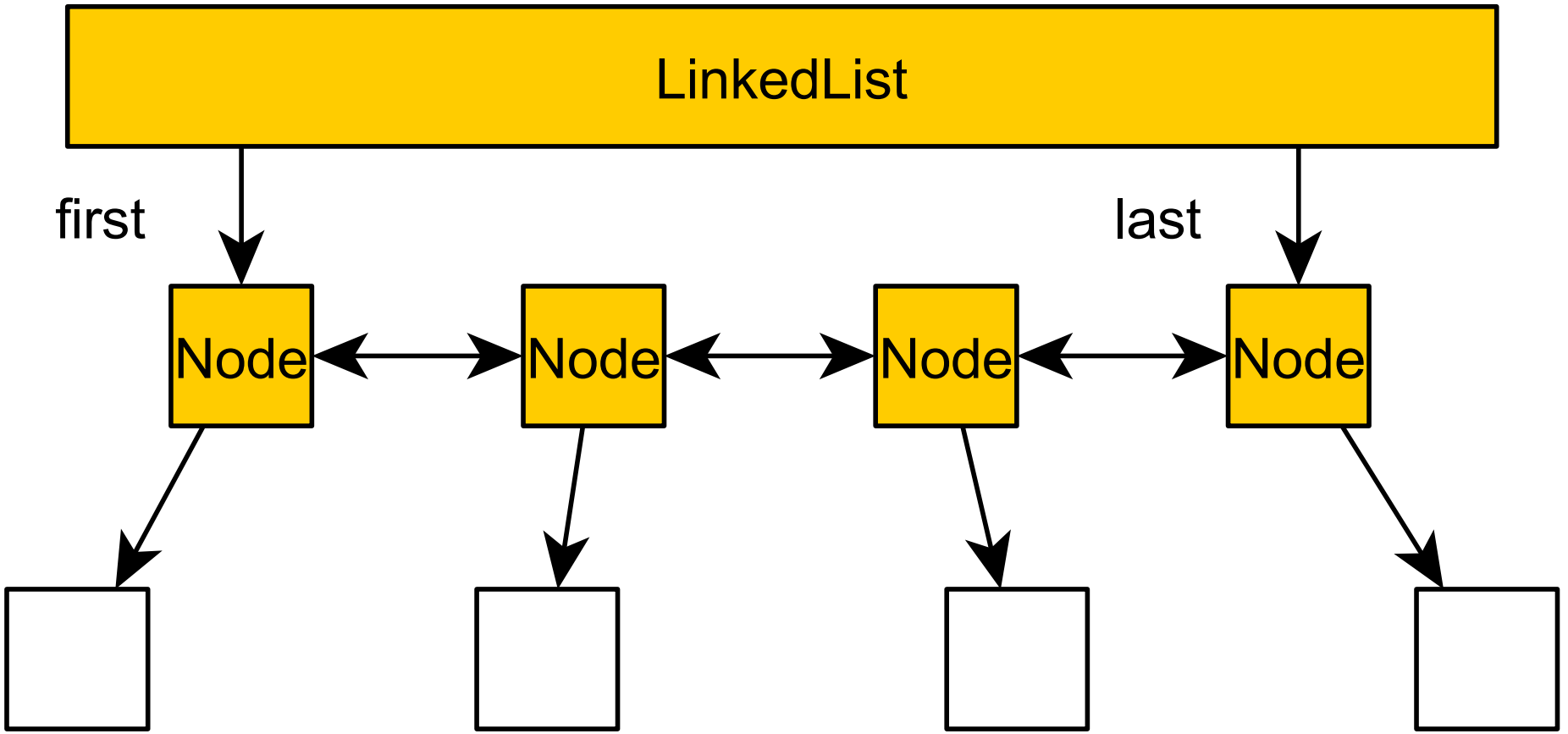
Структуры данных в Java



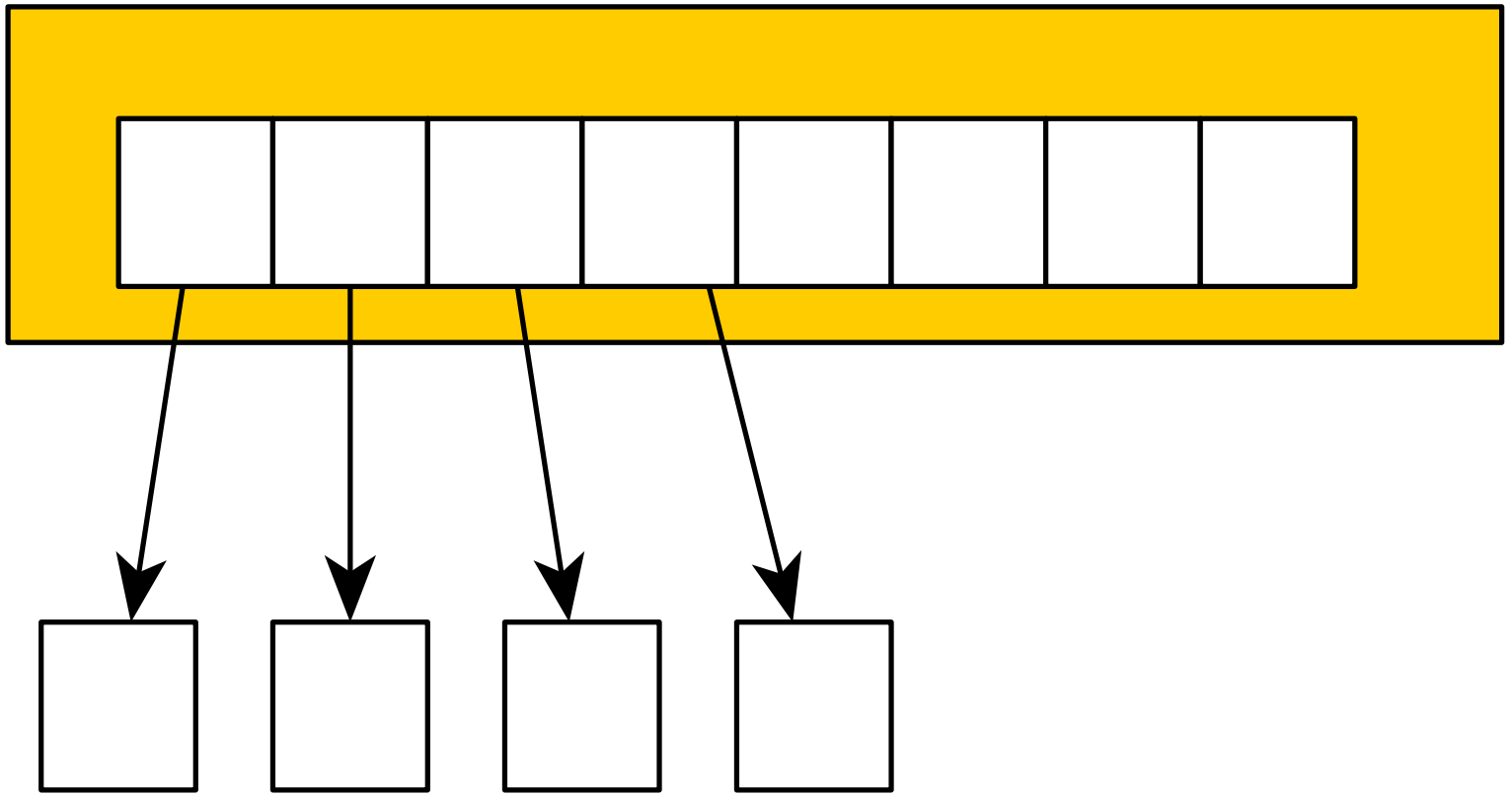
Структуры данных в Java



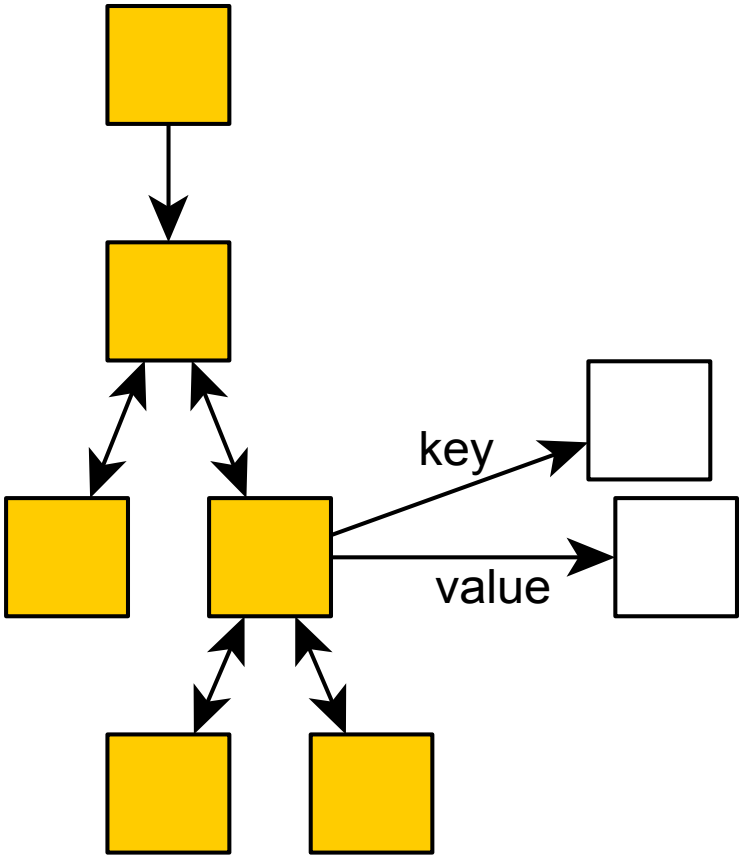
Структуры данных в Java



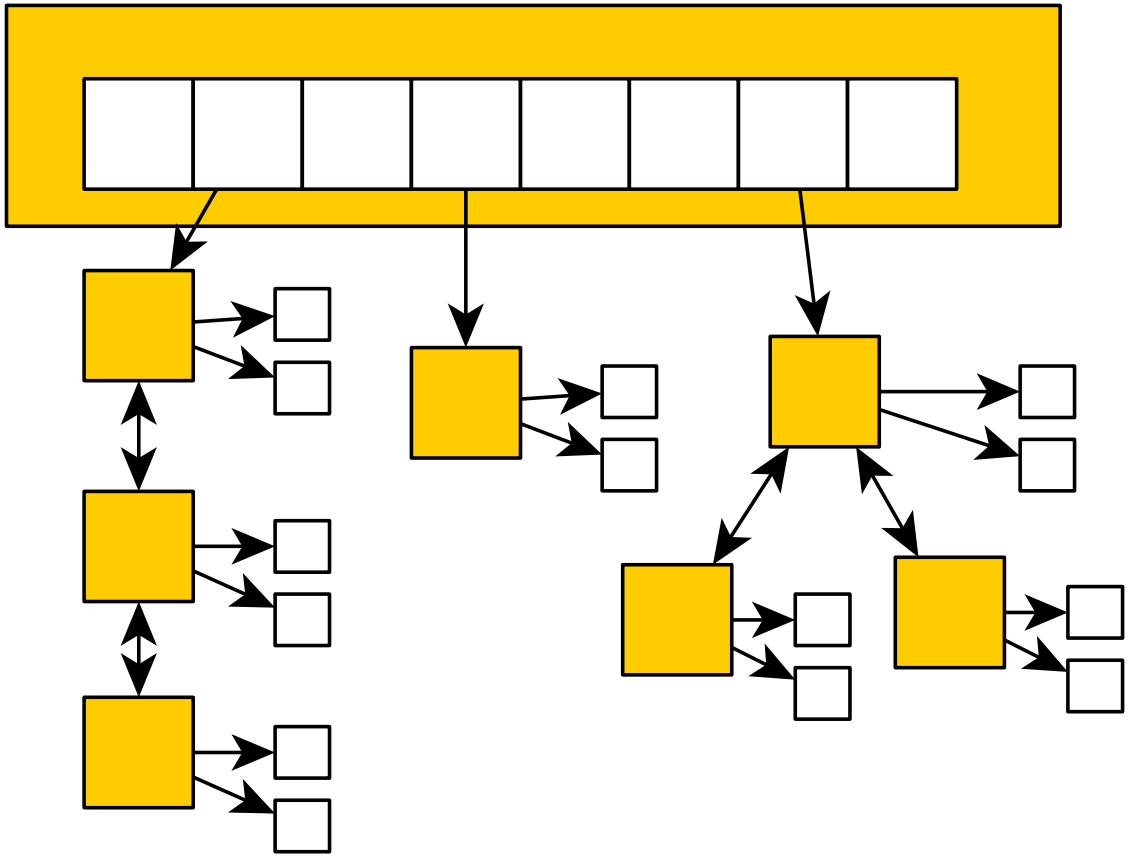
Структуры данных в Java



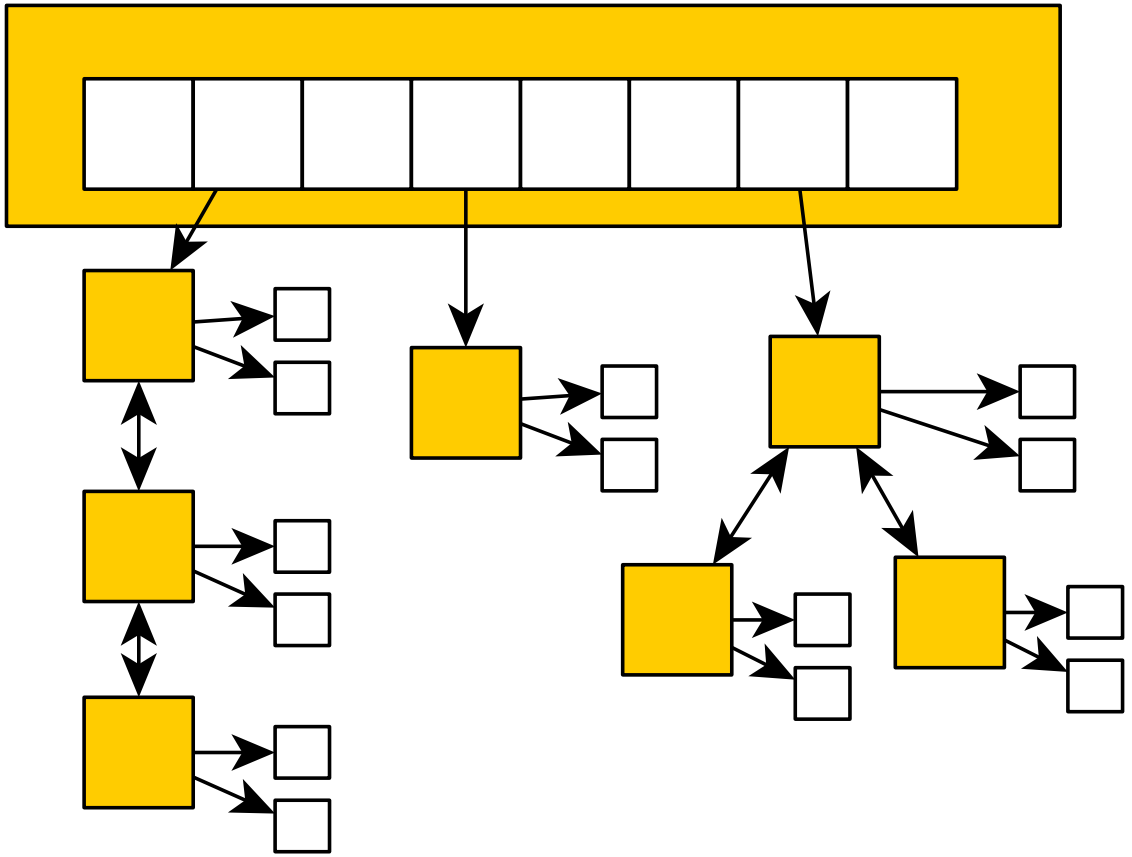
Структуры данных в Java



Структуры данных в Java

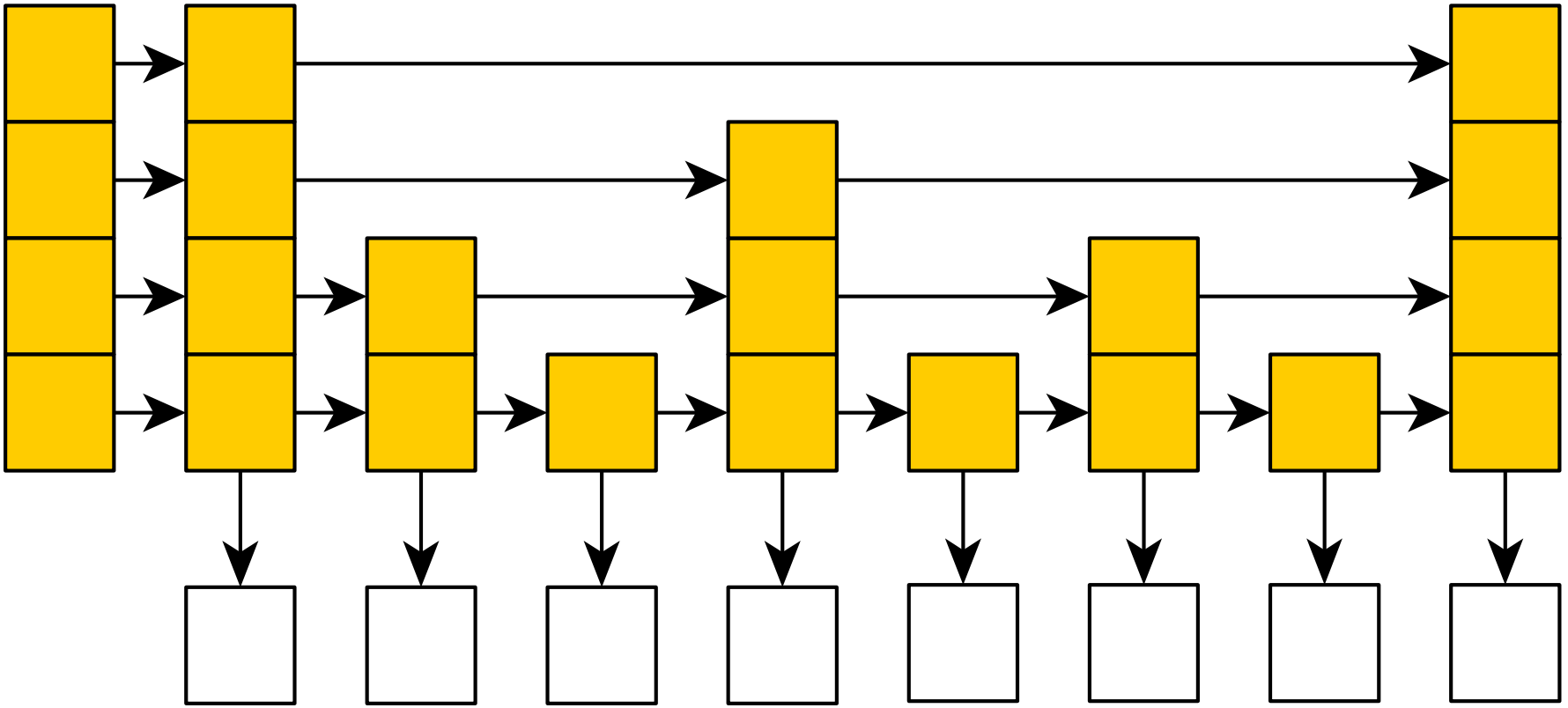


Структуры данных в Java

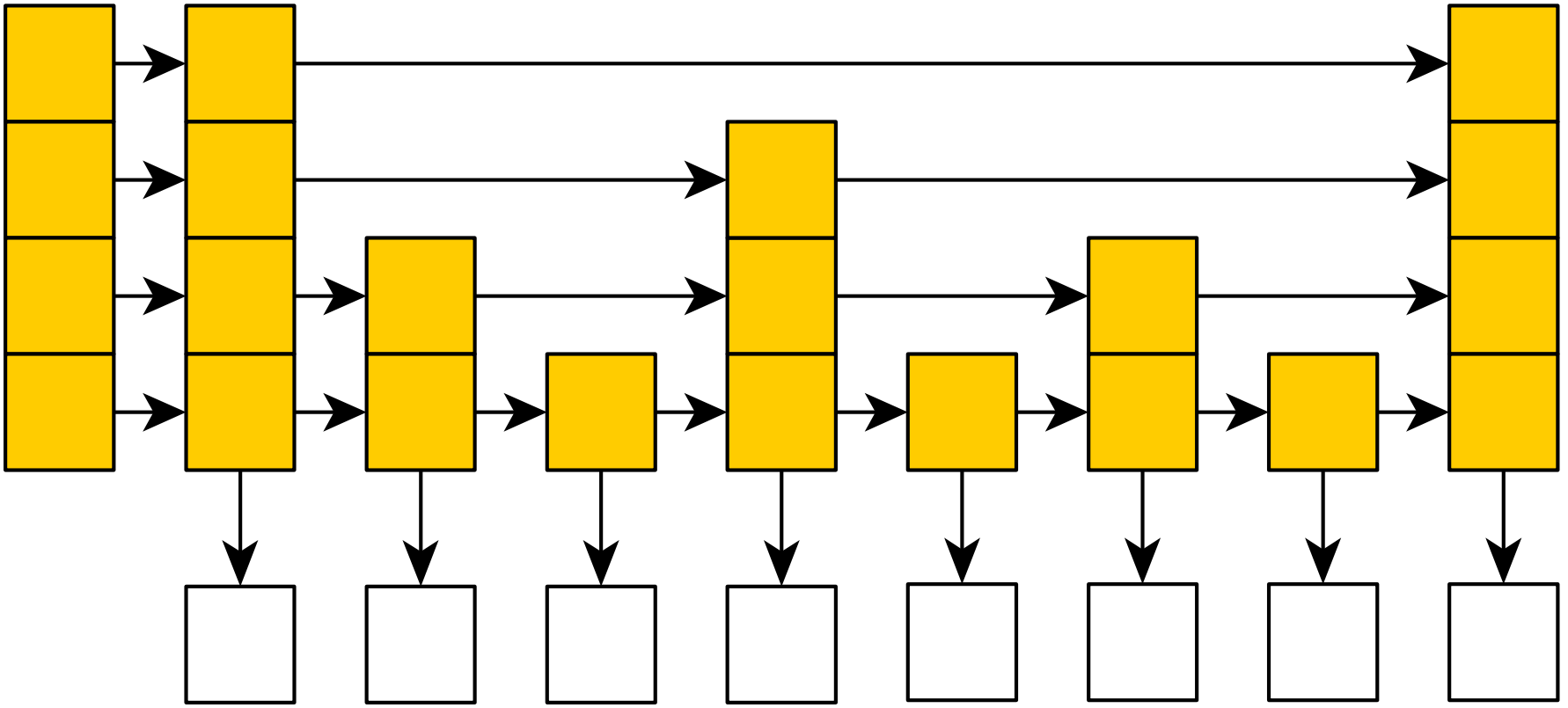


HashMap

Структуры данных в Java

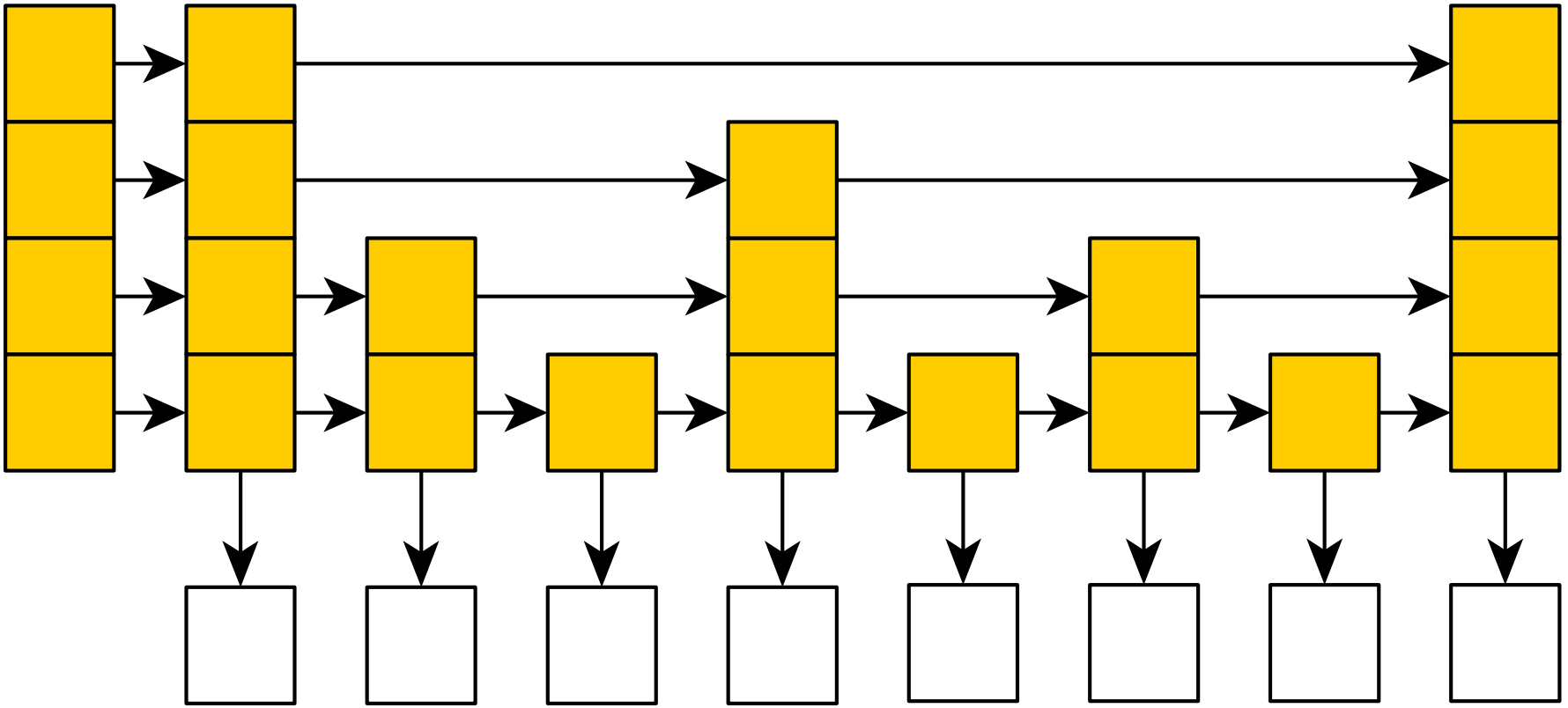


Структуры данных в Java



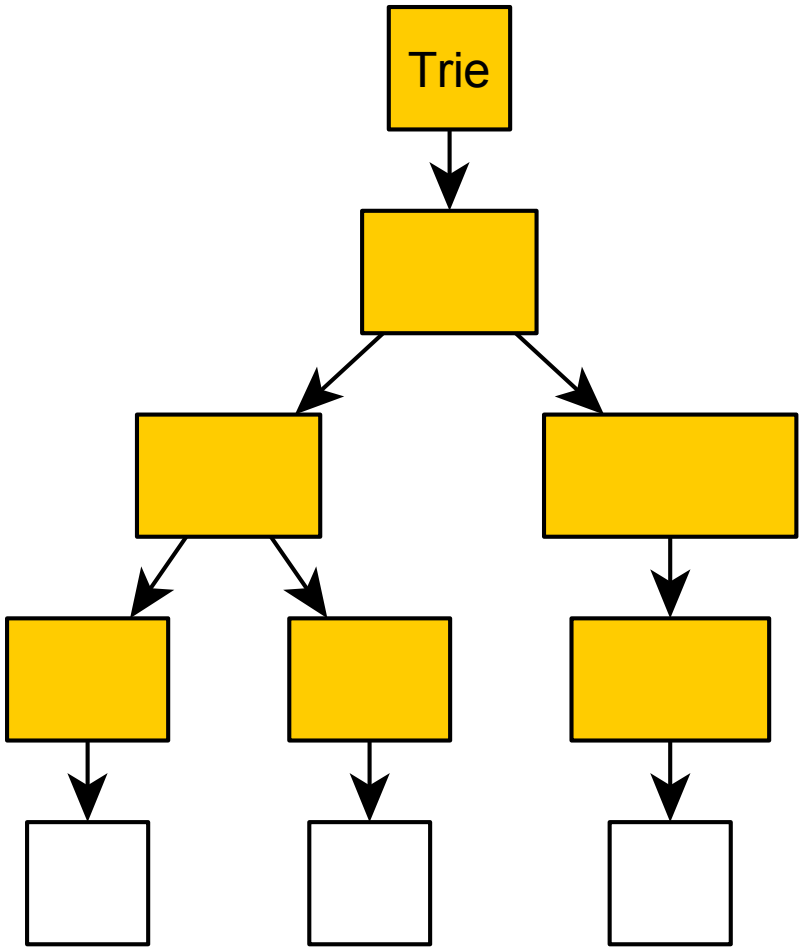
ConcurrentSkipListSet

Структуры данных в Java



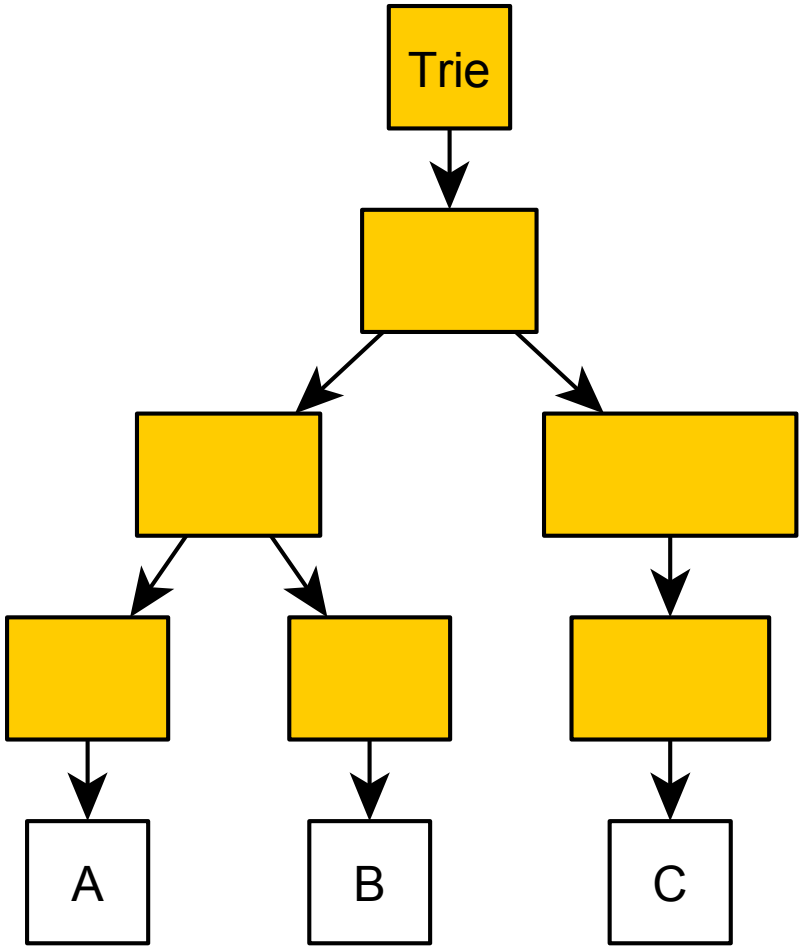
ConcurrentSkipListSet

Trie



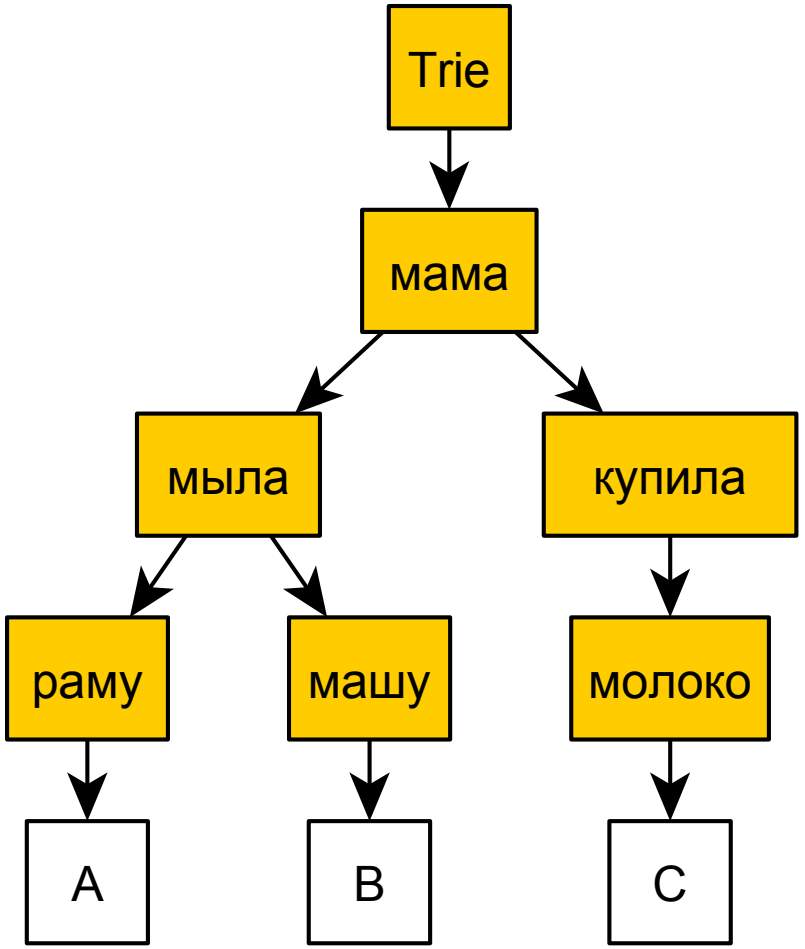
- Trie – дерево

Trie



- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:
  - (ключ$_1$) → A
  - (ключ$_2$) → B
  - (ключ$_3$) → C

Trie



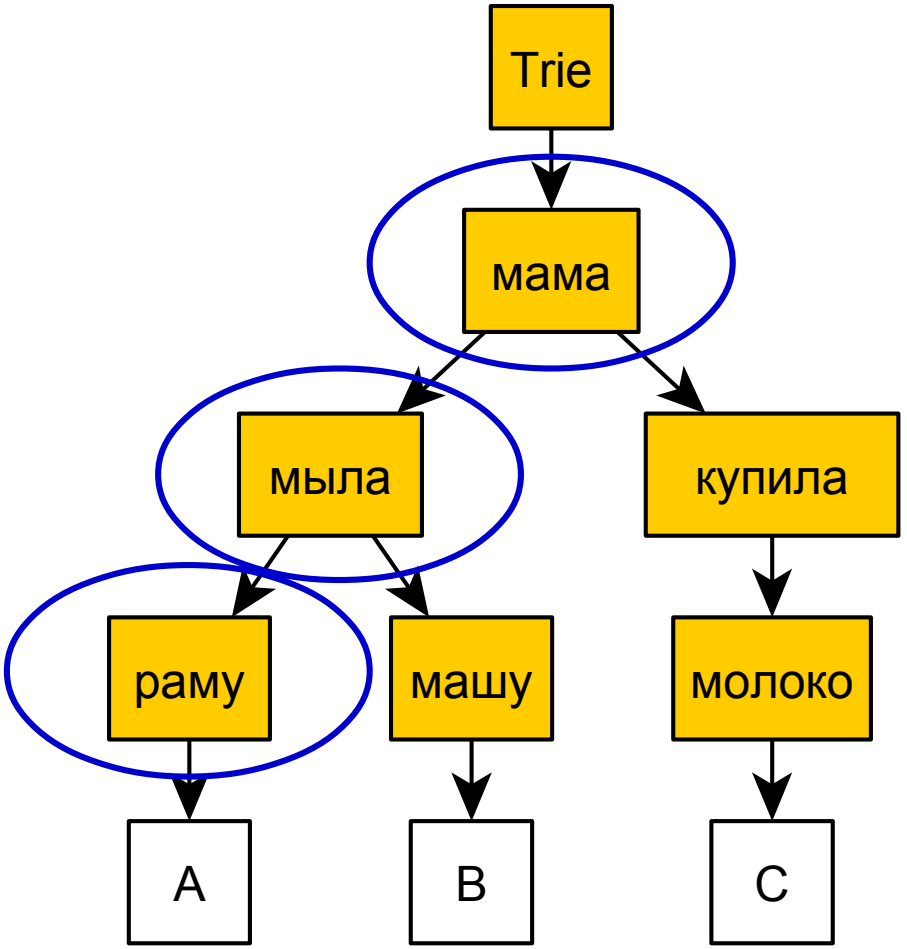
- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → A

«мама мыла машу» → B

«мама купила молоко» → C

Trie



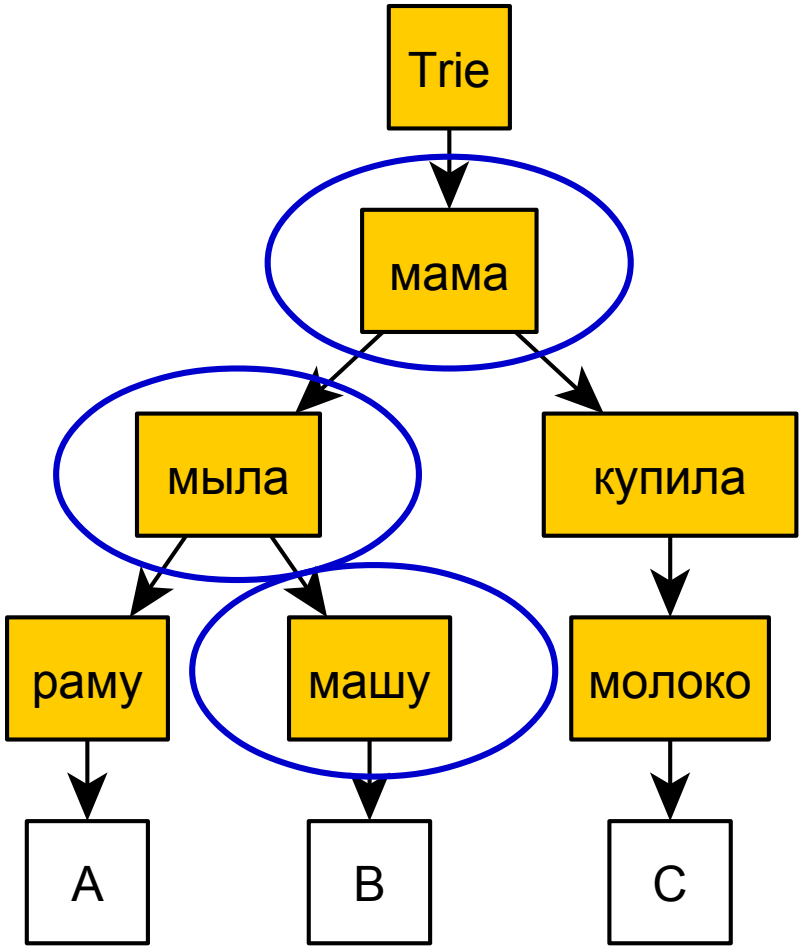
- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → A

«мама мыла машу» → B

«мама купила молоко» → C

Trie



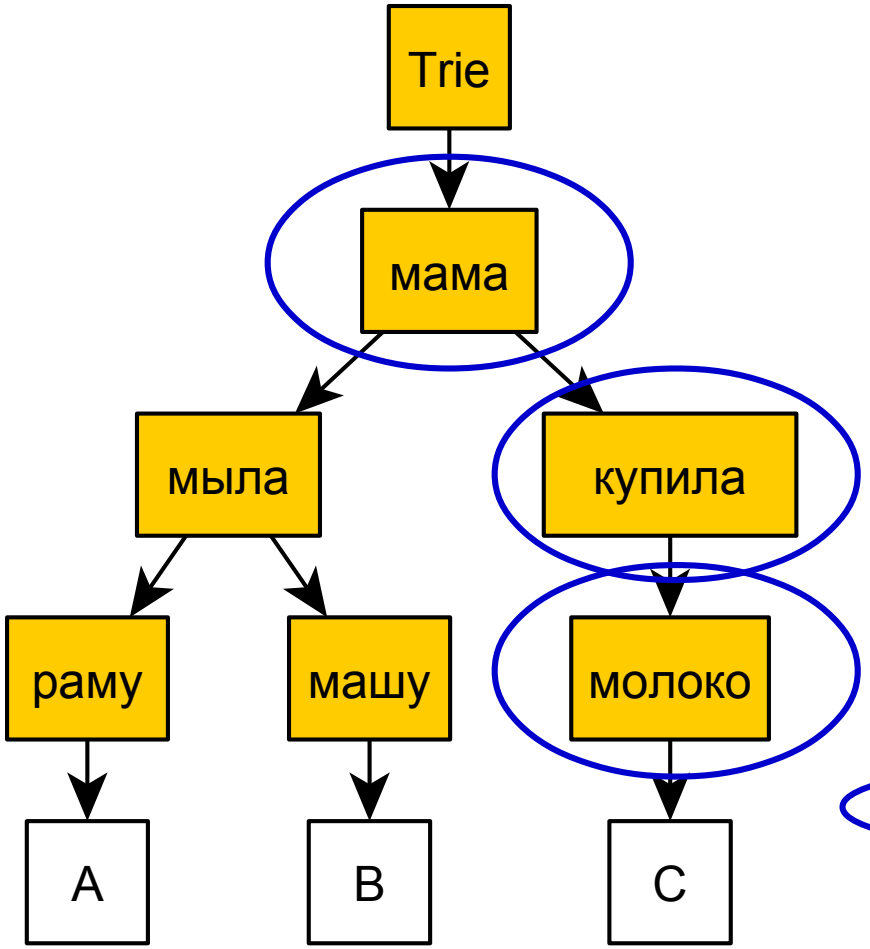
- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → A

«мама мыла машу» → B

«мама купила молоко» → C

Trie



- Trie – дерево
- Trie – структура, решающая «задачу о словаре»:

«мама мыла раму» → A

«мама мыла машу» → B

«мама купила молоко» → C

Trie: применения

- Хранение "суффиксных деревьев"
  - когда ключи имеют общие префиксы
- регулярные выражения
- "нечёткий" поиск
- См. например:
  - *Jetty-9 goes fast with Mechanical Sympathy. Look ahead Trie.*

    `https://webtide.com/jetty-9-goes-fast-with-mechanical-sympathy/`

Trie

```
class Trie<V> {
  Node<V> root = new Node<>();
}
class Node<V> {
  Map<String, Node<V>> children = new HashMap<>();
  String keyPart;
  V value;
}
```

Рейтинг авторов русской Википедии

WIKIPEDIA
The Free Encyclopedia

Рейтинг авторов русской Википедии

- Есть много разных рейтингов:
  - количество начатых статей ("первая правка")
  - количество правок в статьях
  - "стаж"

Рейтинг авторов русской Википедии

- Есть много разных рейтингов:
  - количество начатых статей ("первая правка")
  - количество правок в статьях
  - "стаж"

- Не отражают "качество правки"
- Не отражают популярность статей

Рейтинг авторов русской Википедии

- Построить рейтинг самых читаемых авторов русской Википедии в 2013 году

Рейтинг авторов русской Википедии

- Построить рейтинг самых читаемых авторов русской Википедии в 2013 году
  - пропорционально авторству всего текста в статье, а не факта создания статей

Рейтинг авторов русской Википедии

- Построить рейтинг самых читаемых авторов русской Википедии в 2013 году
  - пропорционально авторству всего текста в статье, а не факта создания статей
  - 1 балл за каждое посещение статьи в 2013 году ("хит") если ты 100% автор

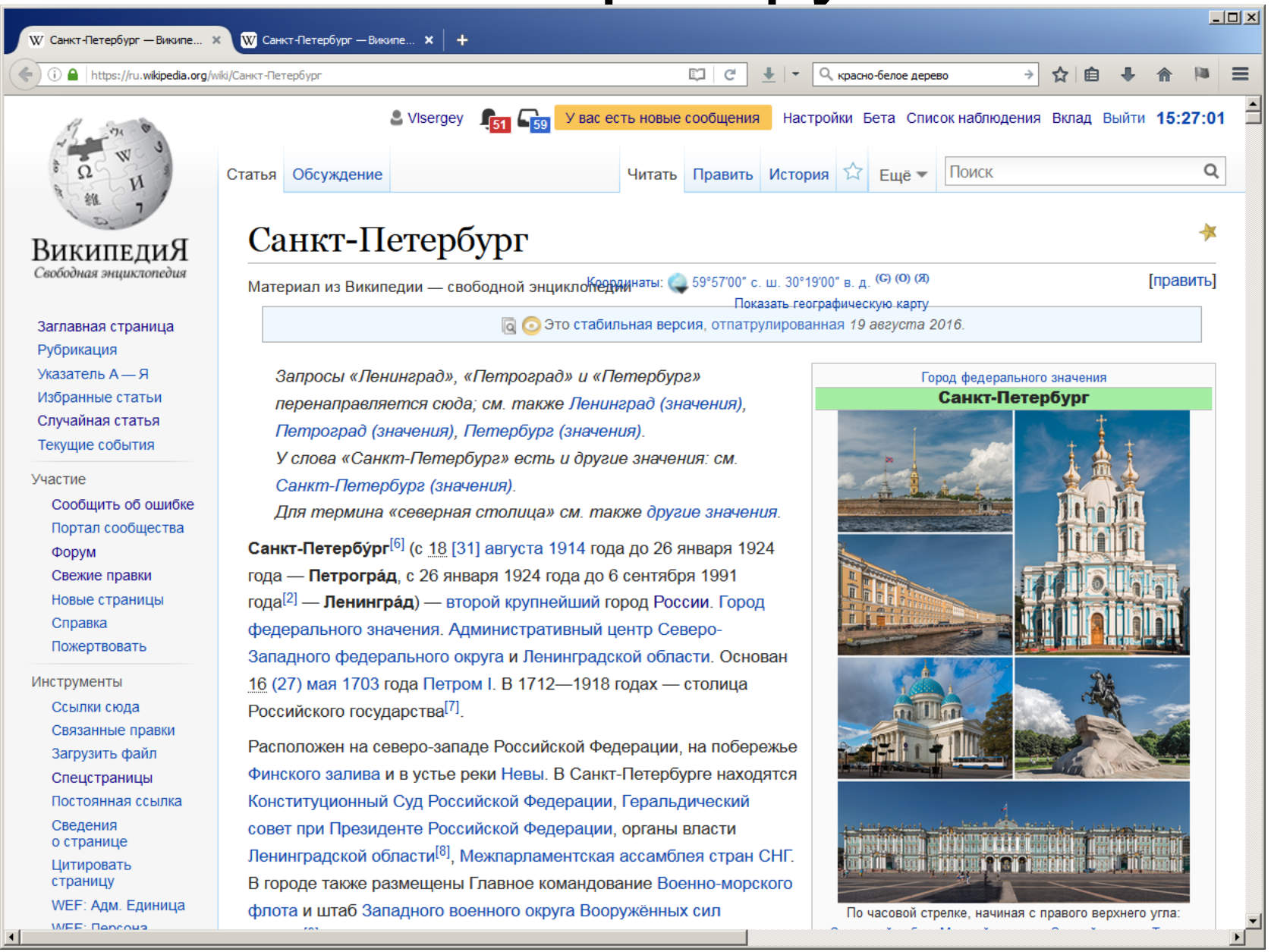
Рейтинг авторов русской Википедии

1. Забираем с stats.wikimedia.org годовую статистику посещений
2. Строим список первых N (N > 1000) популярных статей
3. Для каждой статьи определяем авторство в процентах
4. Считаем итоги

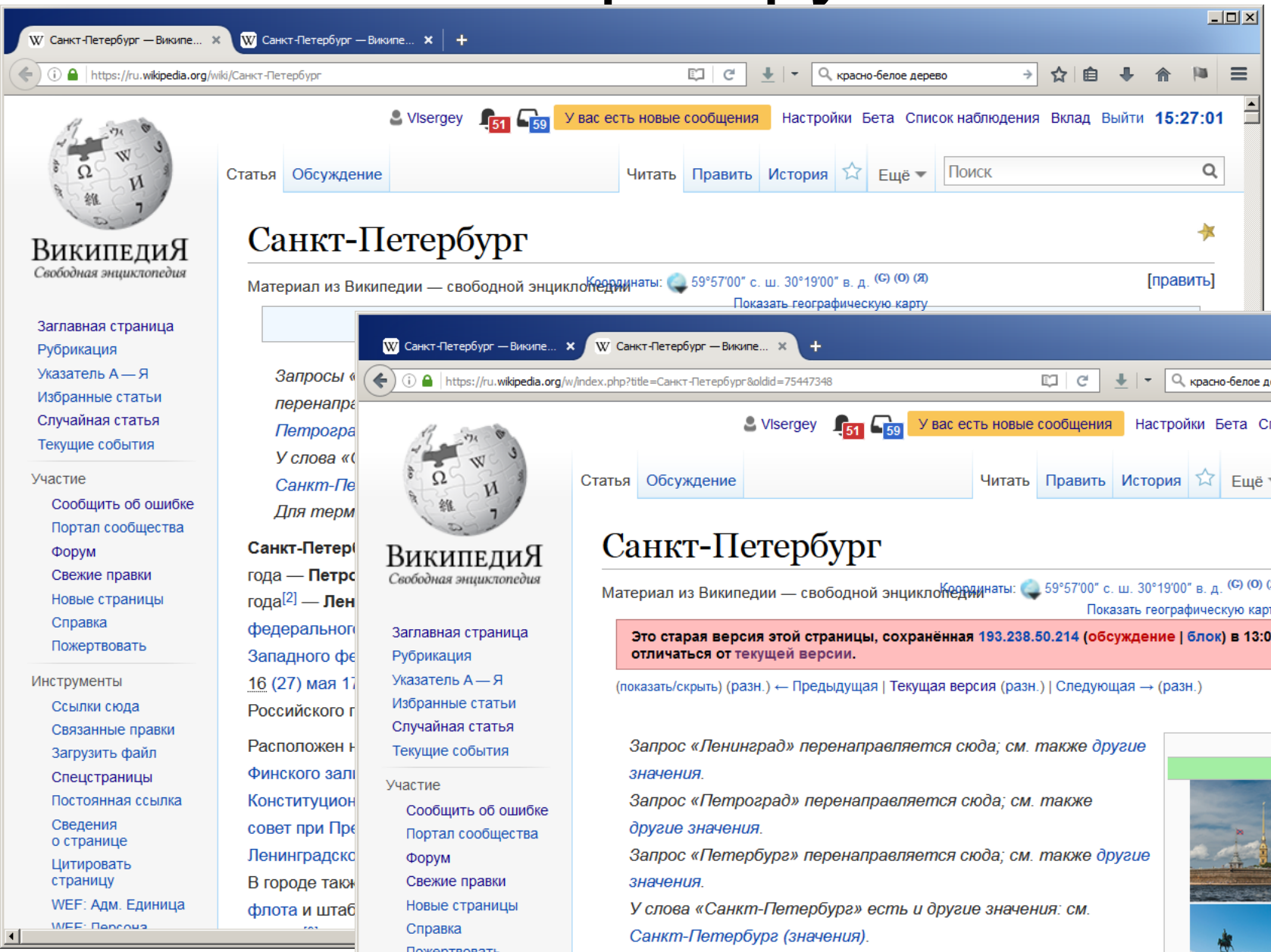
Рейтинг авторов русской Википедии

1. Забираем с stats.wikimedia.org годовую статистику посещений
2. Строим список первых N (N > 1000) популярных статей
3. Для каждой статьи определяем авторство в процентах
4. Считаем итоги

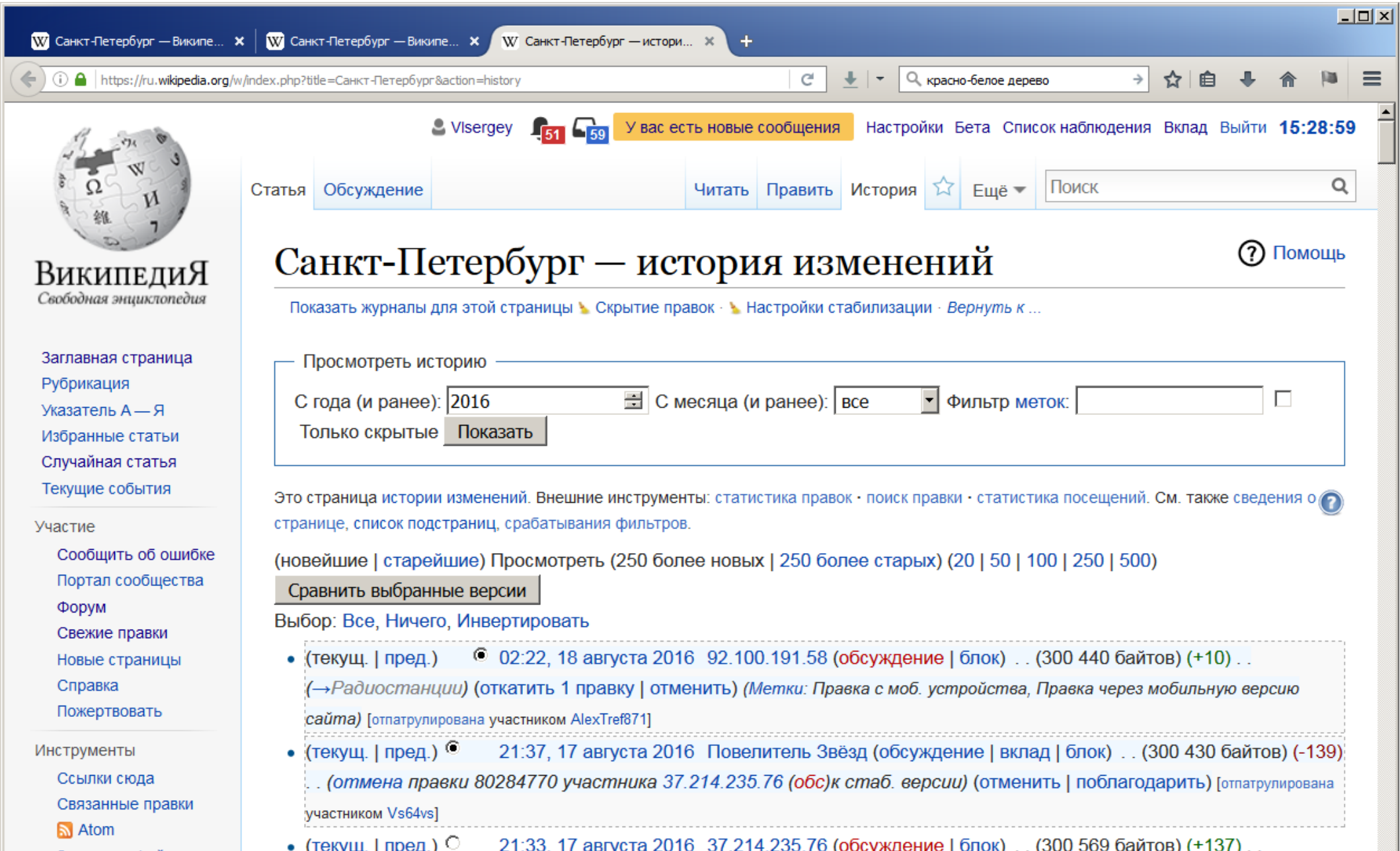
Рейтинг авторов русской Википедии



Рейтинг авторов русской Википедии



Рейтинг авторов русской Википедии



Рейтинг авторов русской Википедии

https://ru.wikipedia.org/w/api.php
?action=query
&prop=revisions
&rvlimit=10
&titles=Санкт-Петербург
&rvprop=timestamp|user|comment
&format=json

Определение авторства текста

- Взять самую первую версию статьи
  (27 декабря 2002 года)
- Узнать его автора
  (участник Theta682)
- Считать, что это автор 100% оригинального текста статьи (на 27 декабря 2002 года)

Определение авторства текста

- Взять следующую версию статьи
  (7 января 2003)
- Найти новые предложения и словосочетания в тексте и пометить их авторство
- Считать, что автор новой версии имеет столько авторства статьи, сколько нового текста в ней

Сравнение двух текстов

- автор А:

- сгущённое молоко — молоко вкусное

- автор Б:

- сгущённое молоко — очень вкусное

Сравнение двух текстов

- построить рейтинг авторов
  - определить авторство текста
    - сравнить тексты между собой
      - ...

Сравнение двух текстов

- построить рейтинг авторов
  - определить авторство текста
    - сравнить тексты между собой
      - ...
      -
      -
      - Trie

Сравнение двух текстов

- построить рейтинг авторов
  - определить авторство текста
    - сравнить тексты между собой
      - алгоритм поиска наибольшей общей подстроки

Сравнение двух текстов

- Автор А:
- сгущённое молоко — молоко вкусное

- Автор Б:
- сгущённое молоко — очень вкусное

- 1. "сгущённое молоко" (автор А)

Сравнение двух текстов

- Автор А:
- ~~сгущённое молоко~~ —молоко вкусное

- Автор Б:
- ~~сгущённое молоко~~ —очень вкусное

- 1. "сгущённое молоко" (автор А)
- 2. "вкусное" (автор А)

Сравнение двух текстов

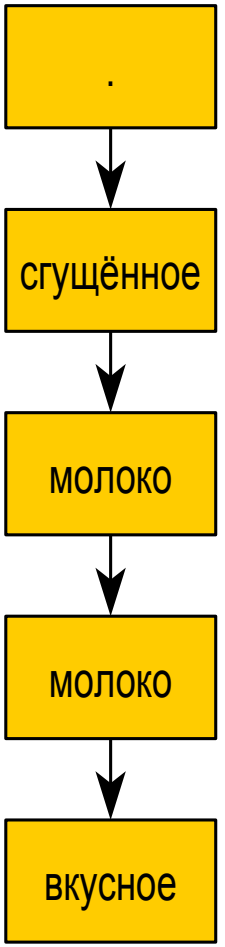
- ~~Автор А:~~
- ~~сгущённое молоко~~ —молоко ~~вкусное~~

- Автор Б:
- ~~сгущённое молоко~~ —очень ~~вкусное~~

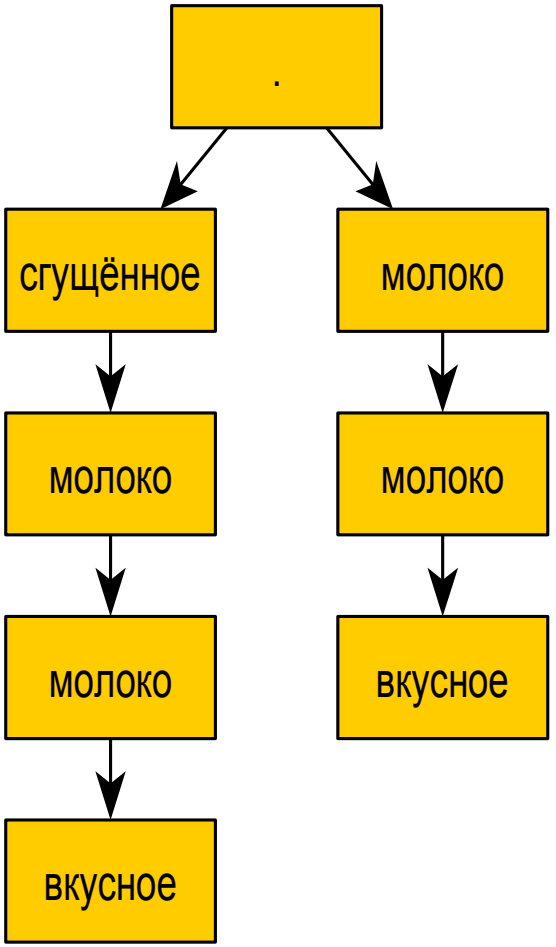
- 1. "сгущённое молоко" (автор А)
- 2. "вкусное" (автор А)

ост.: "очень" (автор Б)

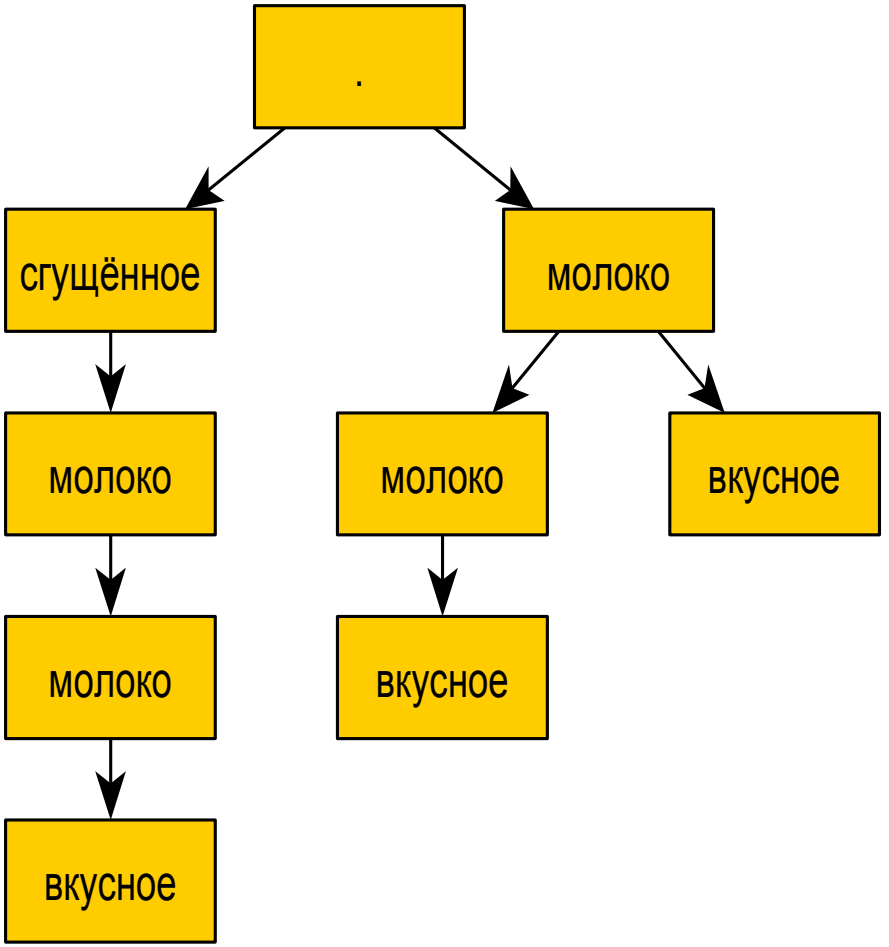
Поиск наибольшей общей подстроки



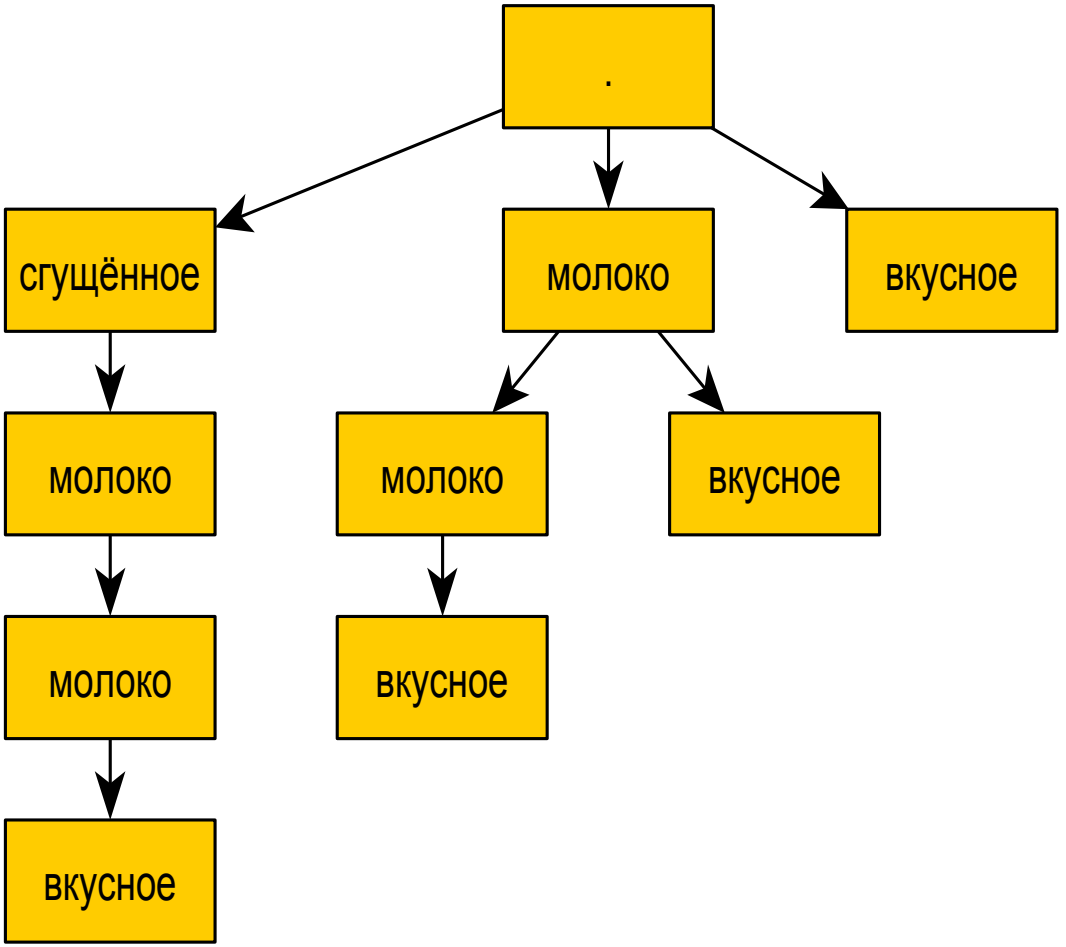
Поиск наибольшей общей подстроки



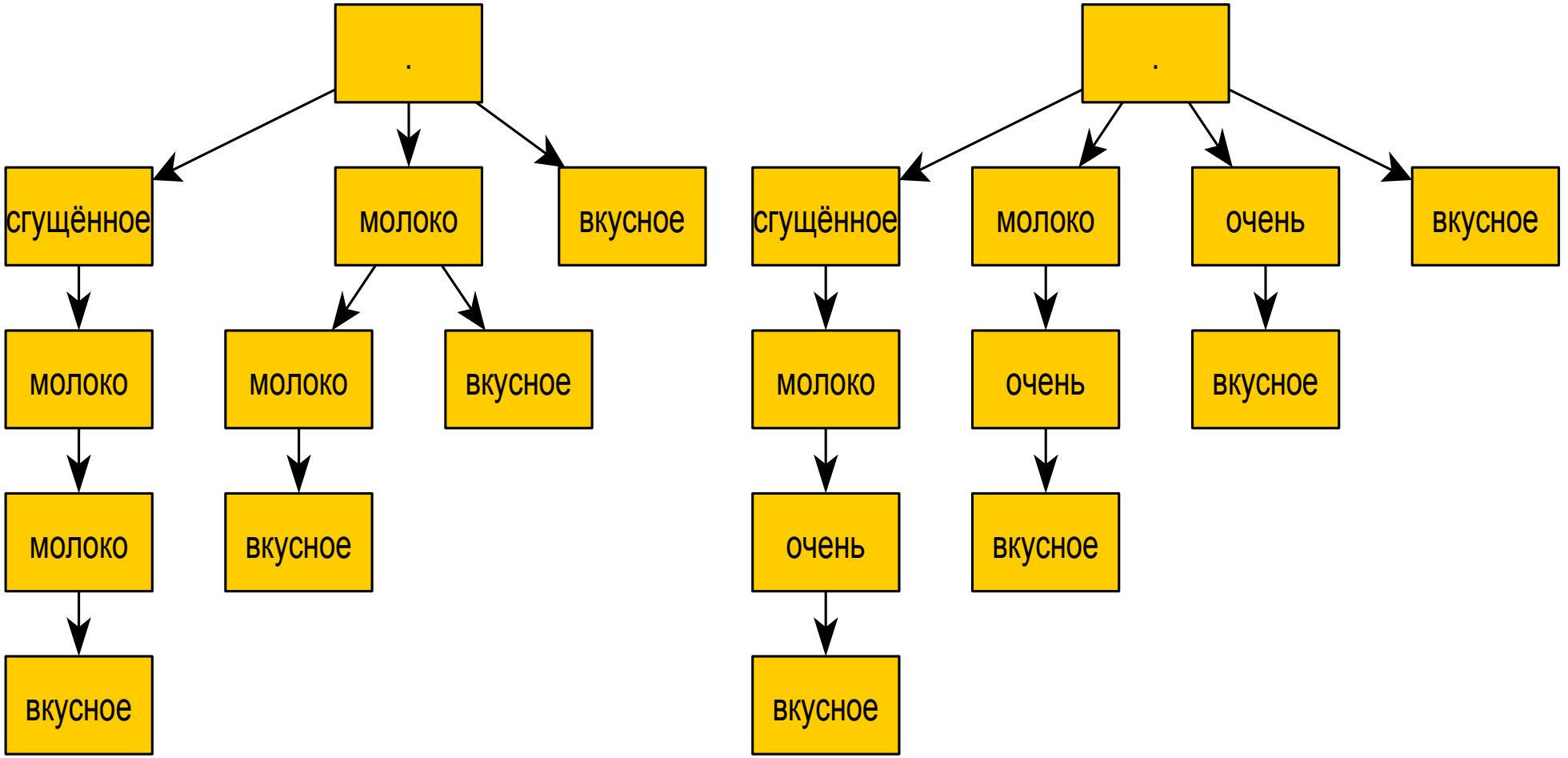
Поиск наибольшей общей подстроки



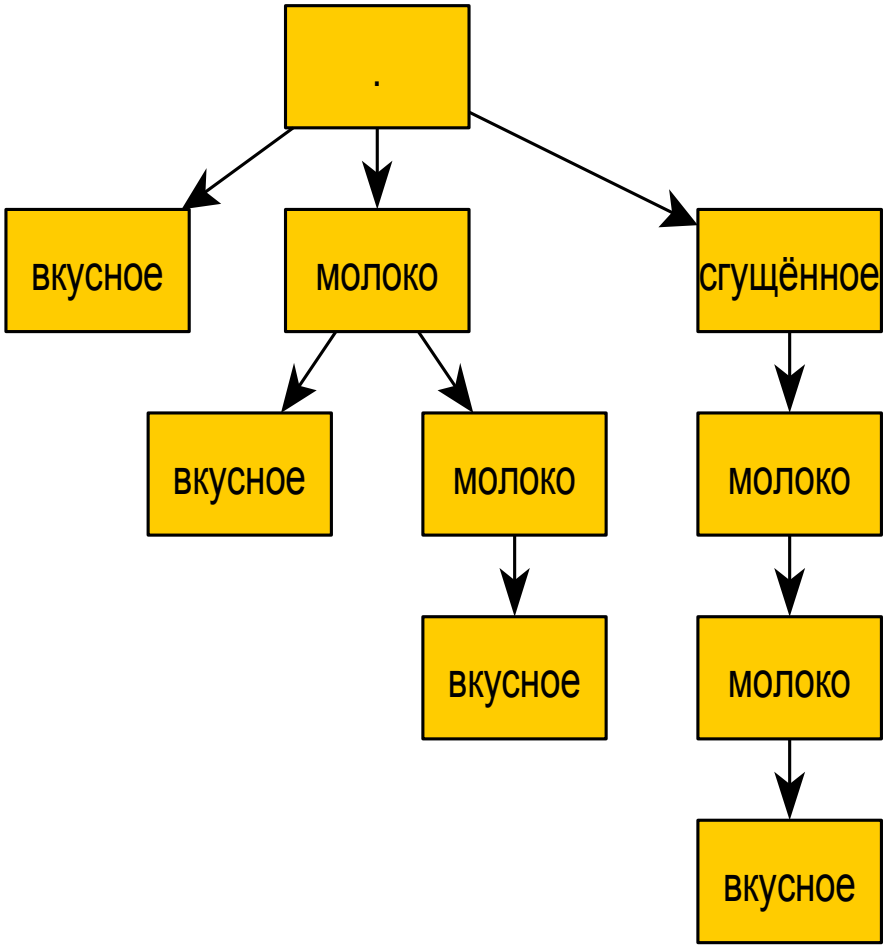
Поиск наибольшей общей подстроки

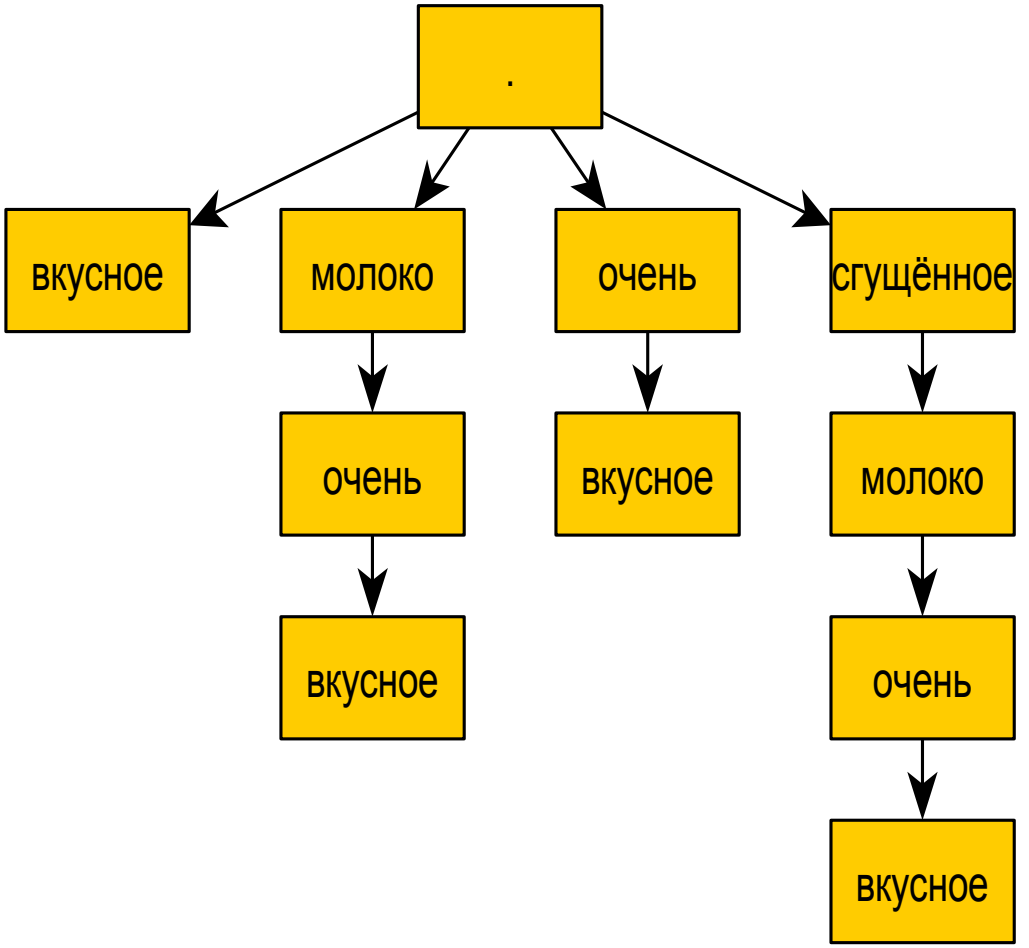


Поиск наибольшей общей подстроки

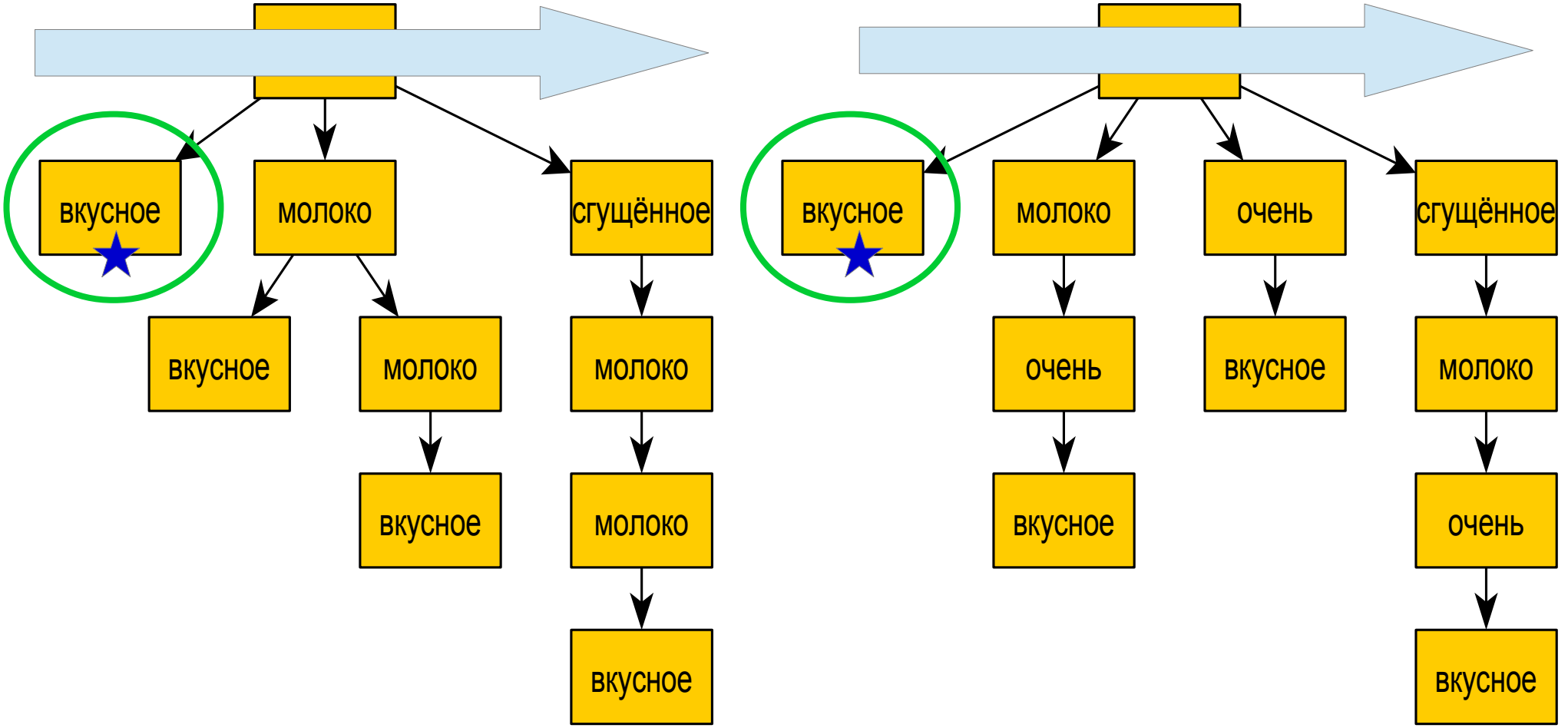


Поиск наибольшей общей подстроки

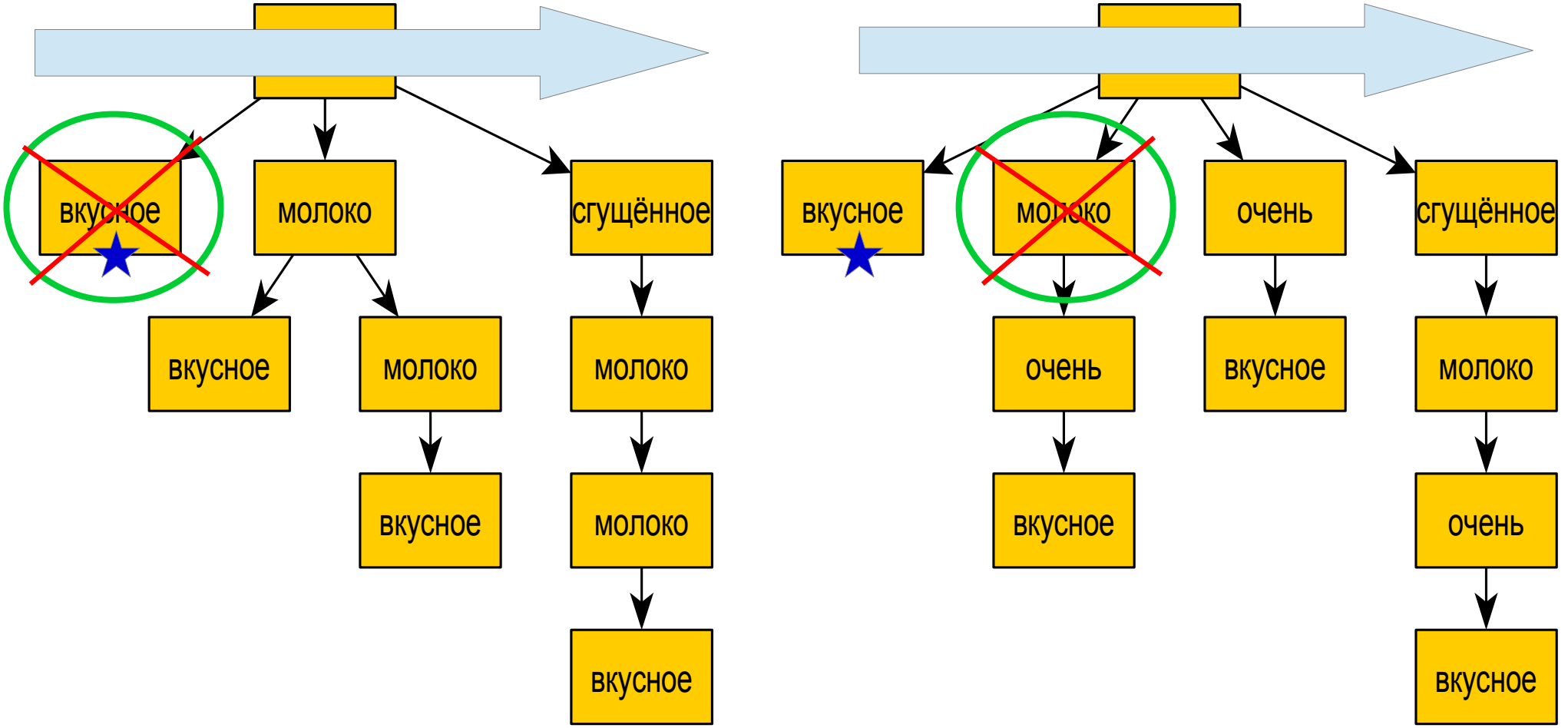




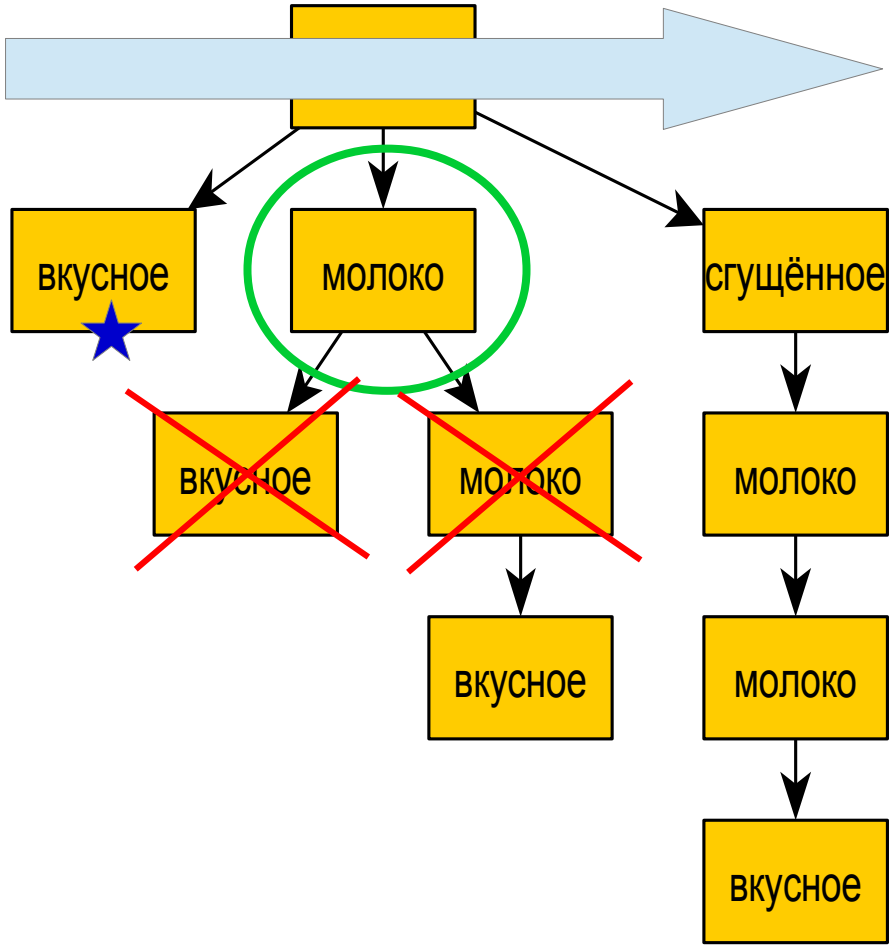
Поиск наибольшей общей подстроки

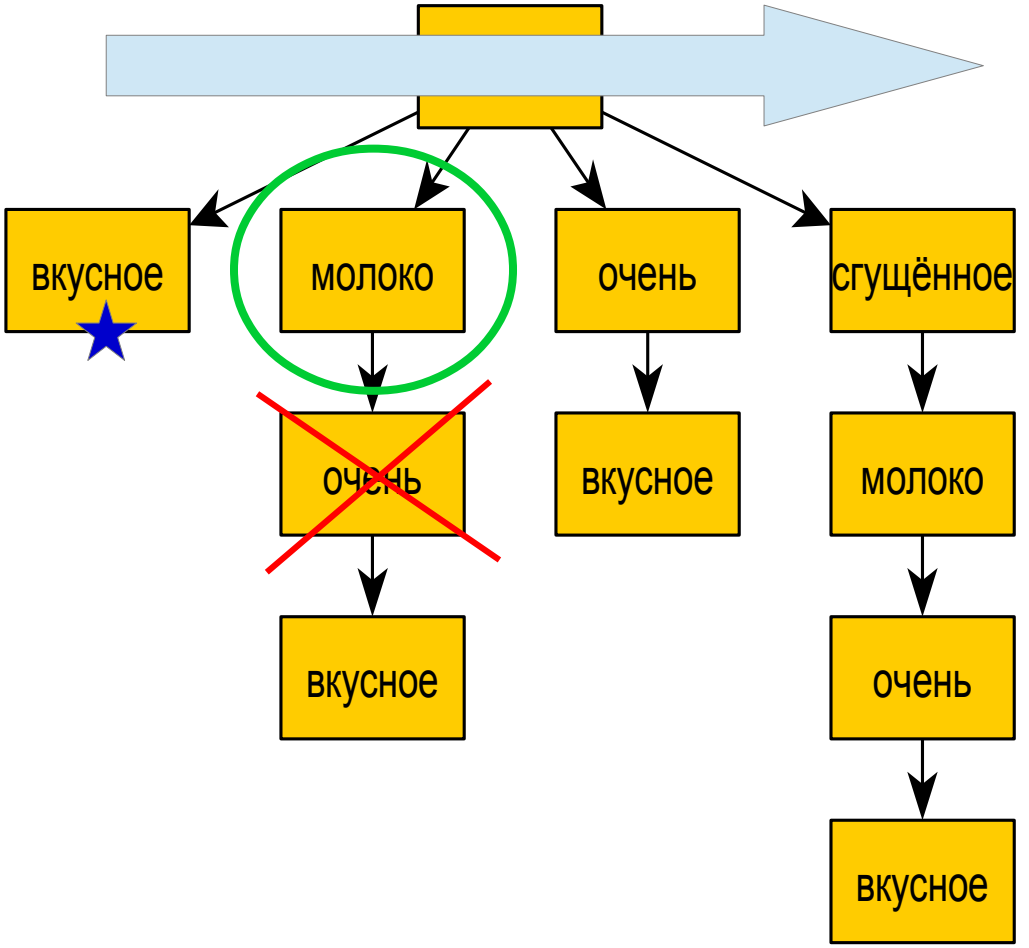


Поиск наибольшей общей подстроки

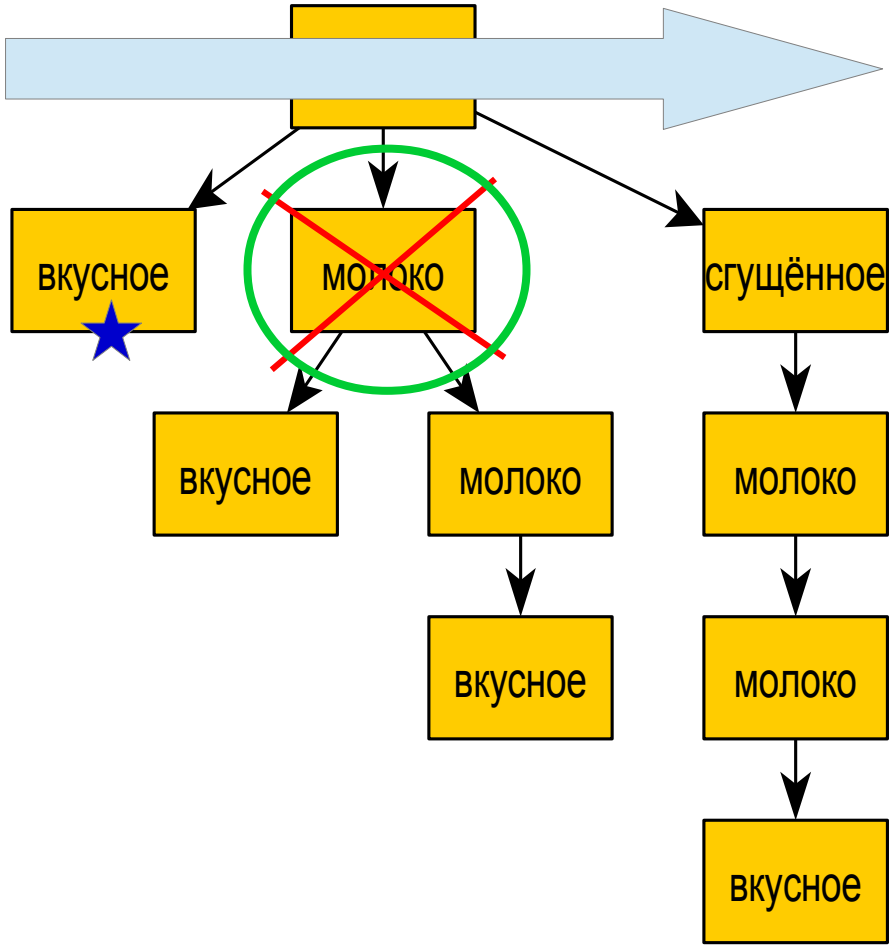


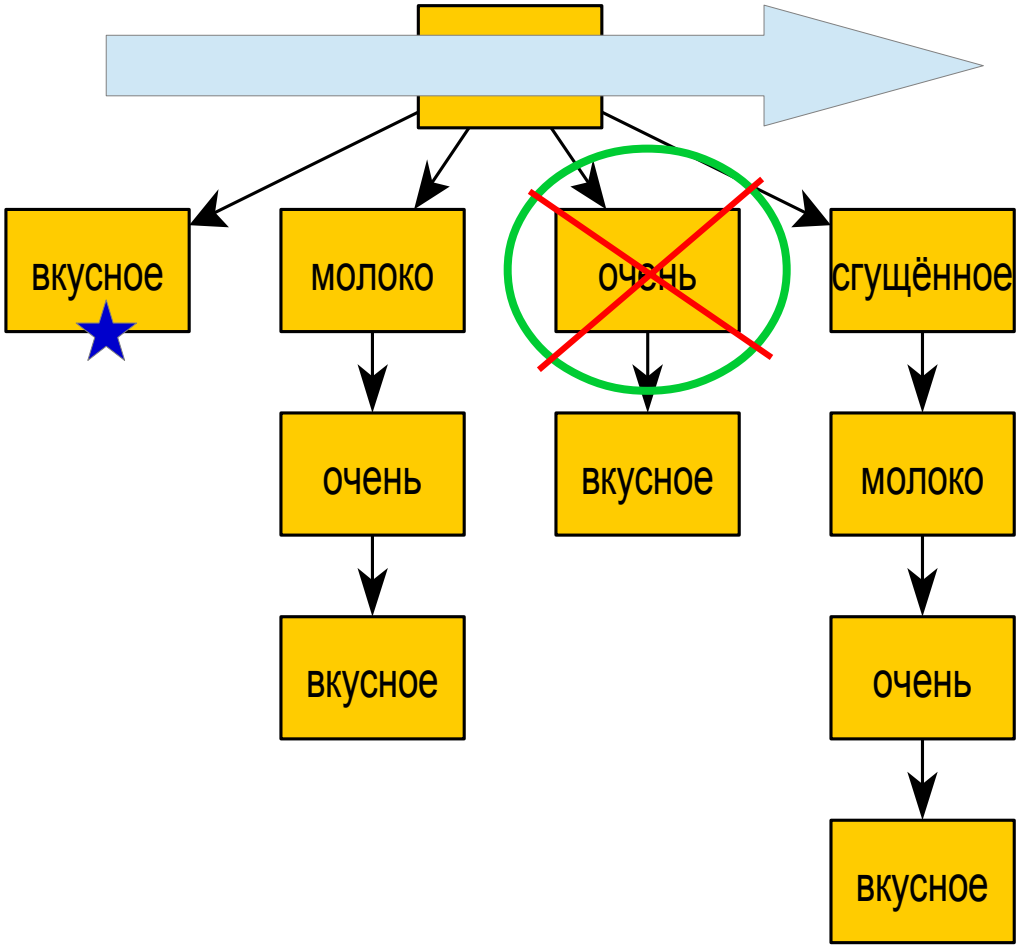
Поиск наибольшей общей подстроки



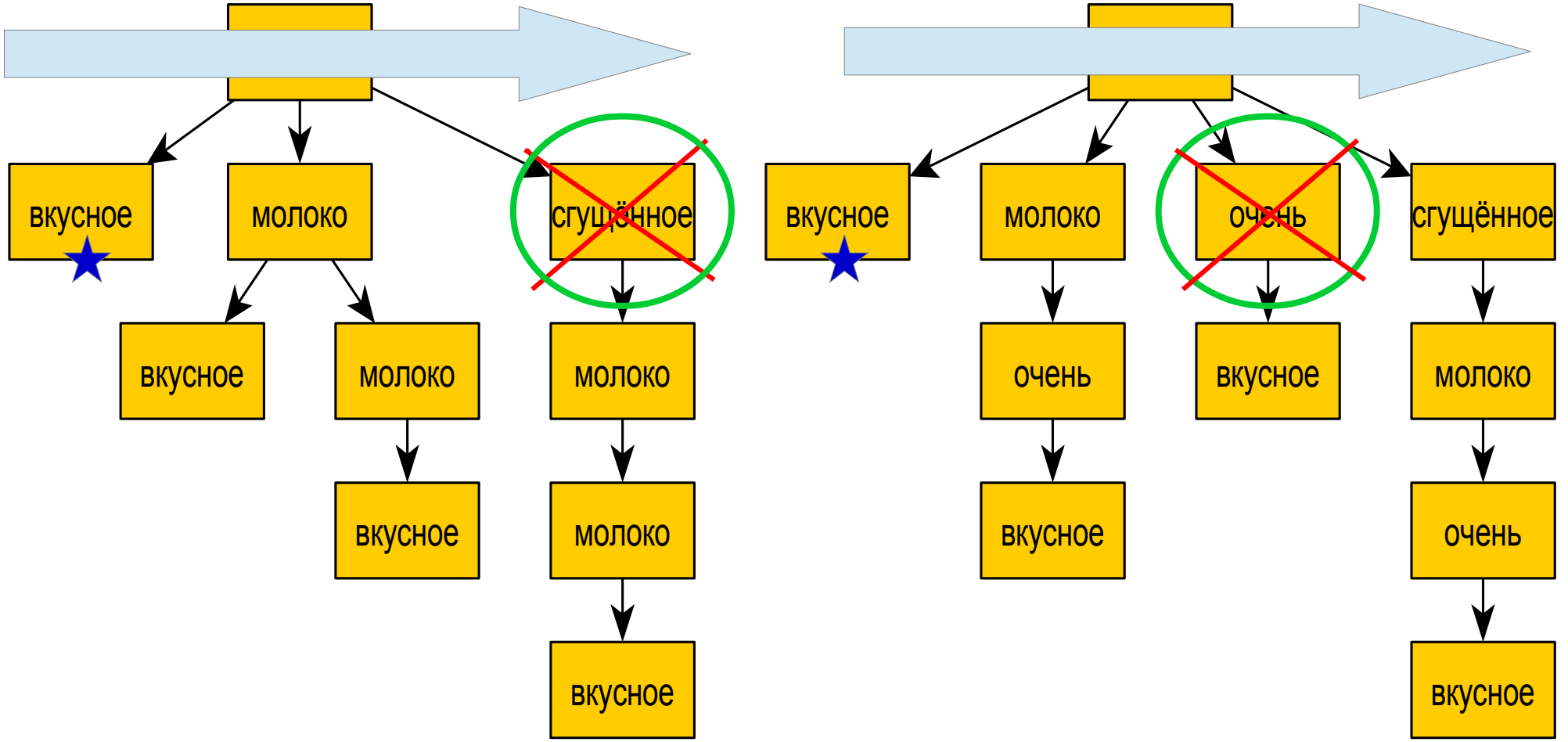


Поиск наибольшей общей подстроки

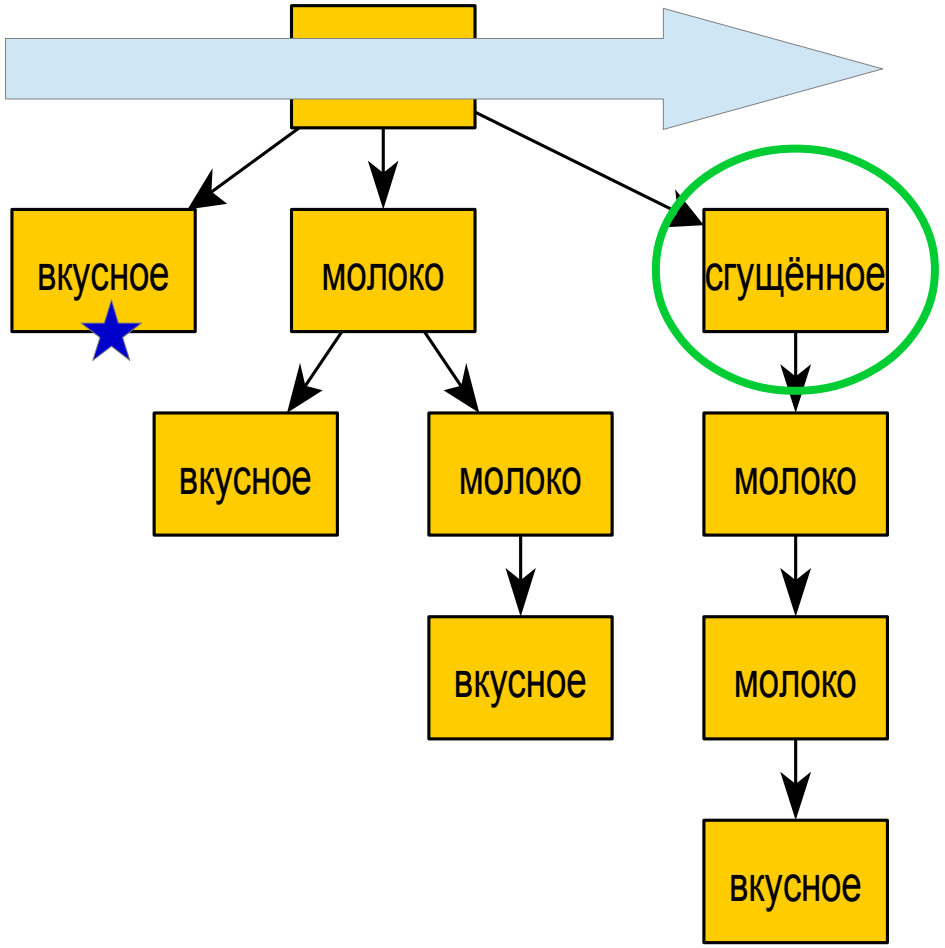


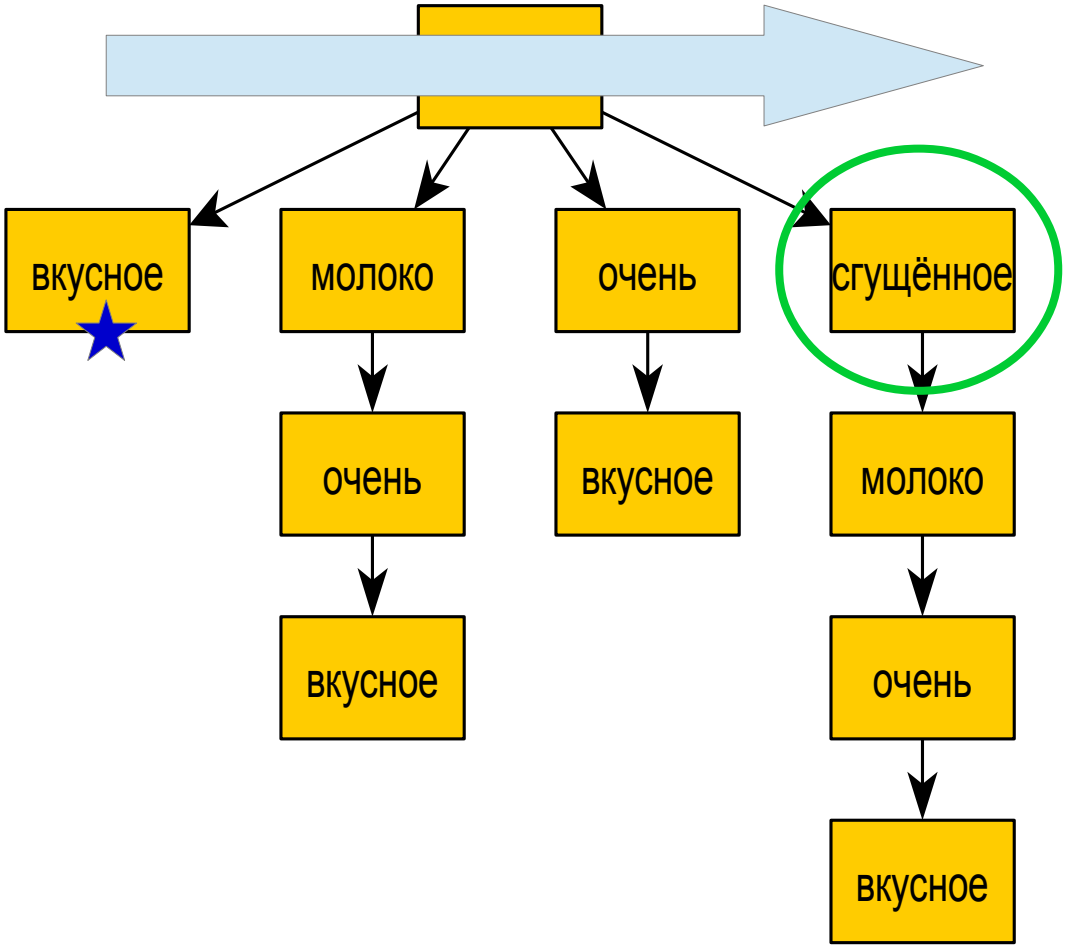


Поиск наибольшей общей подстроки

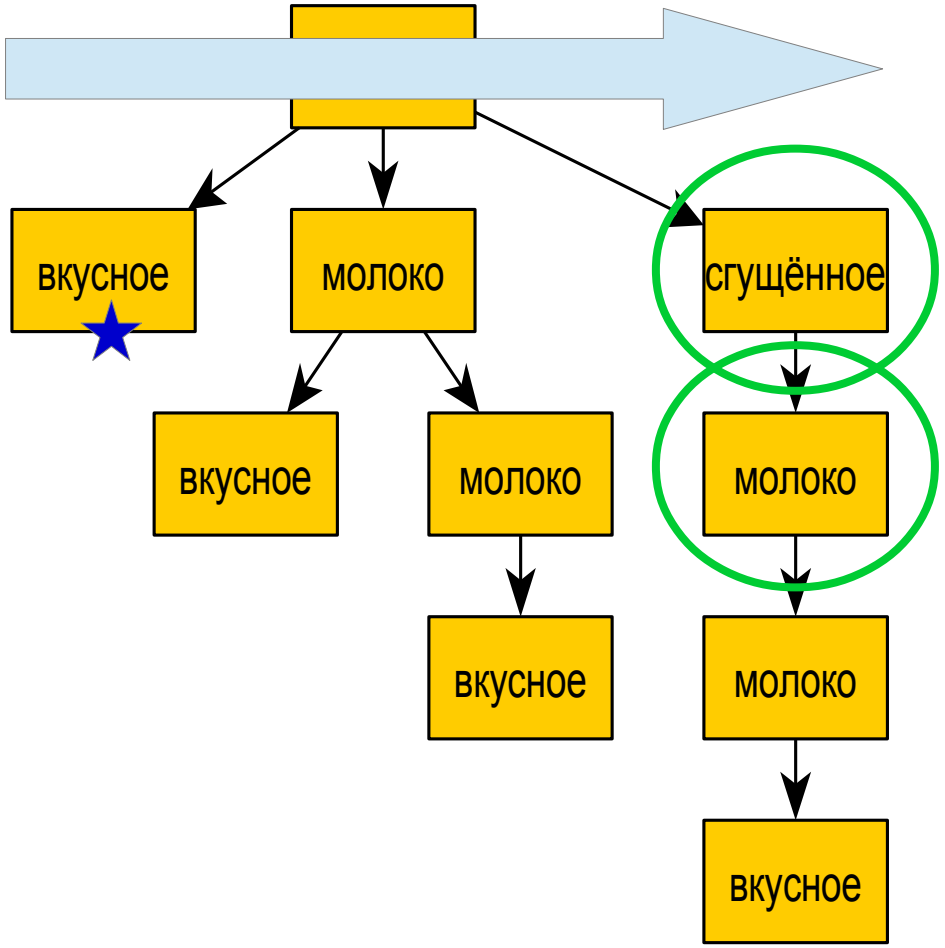


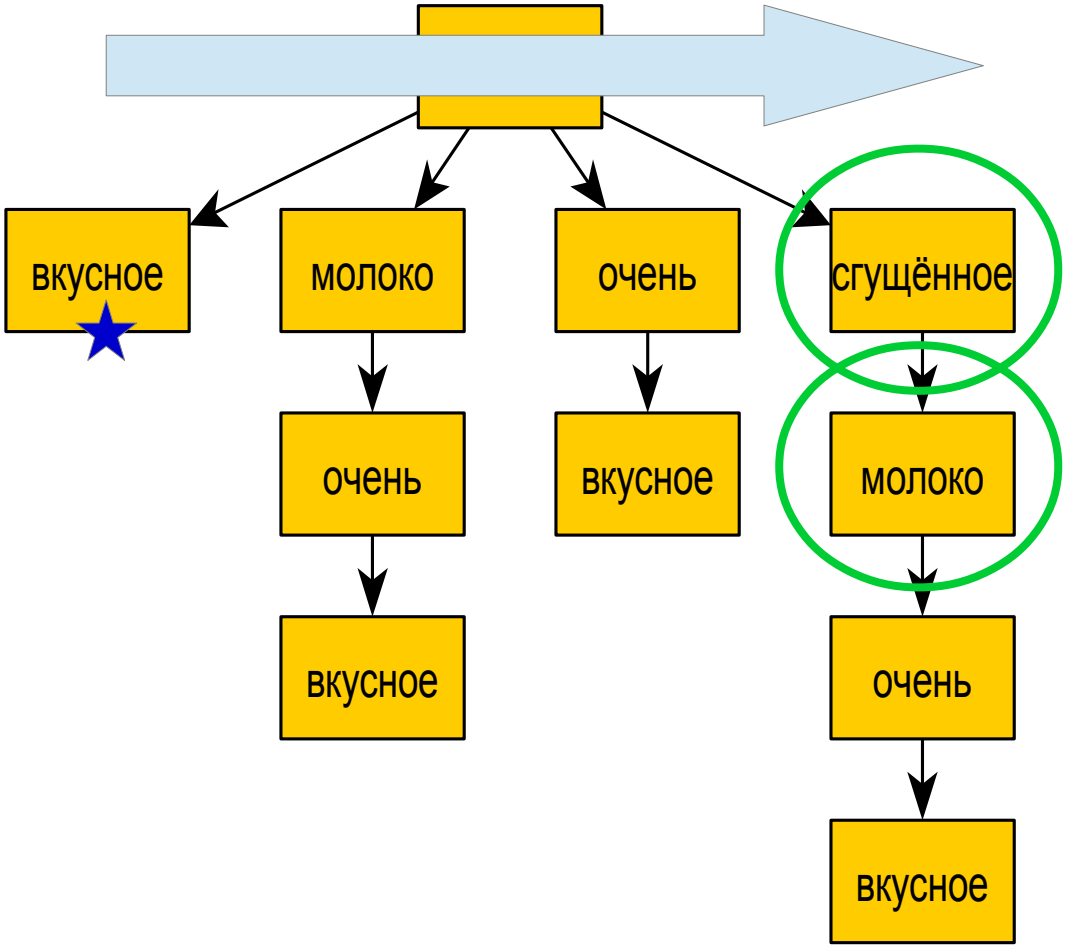
Поиск наибольшей общей подстроки



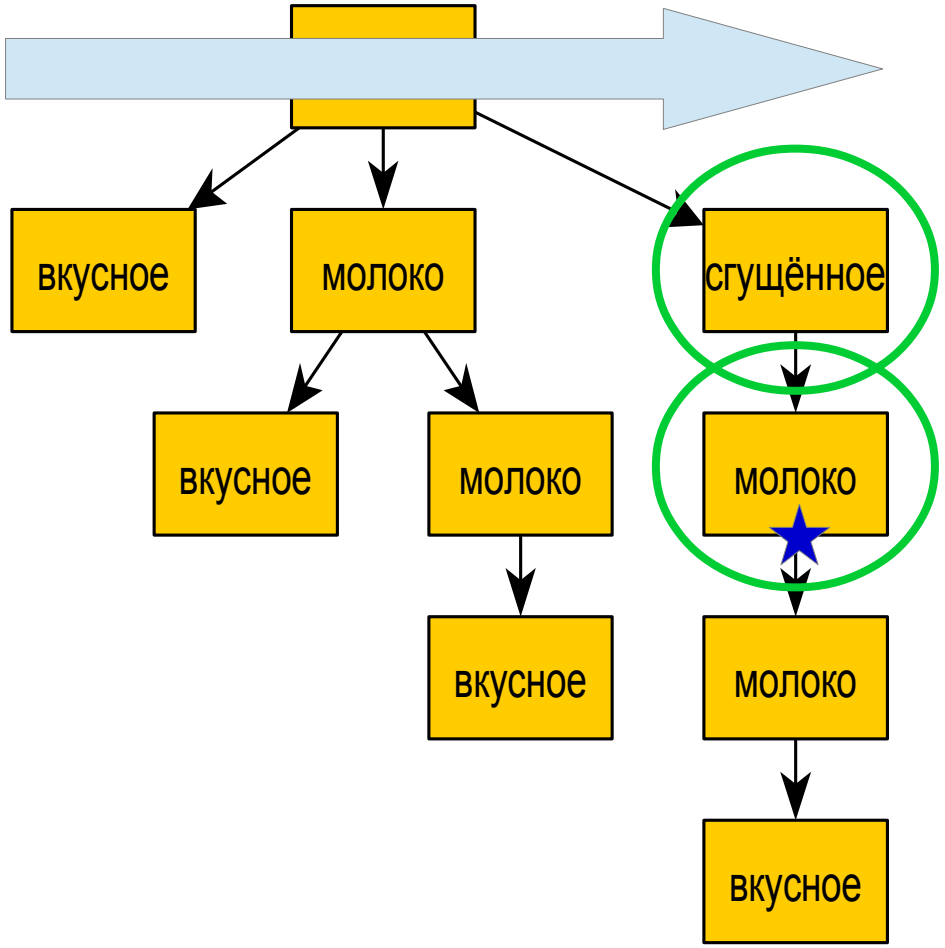


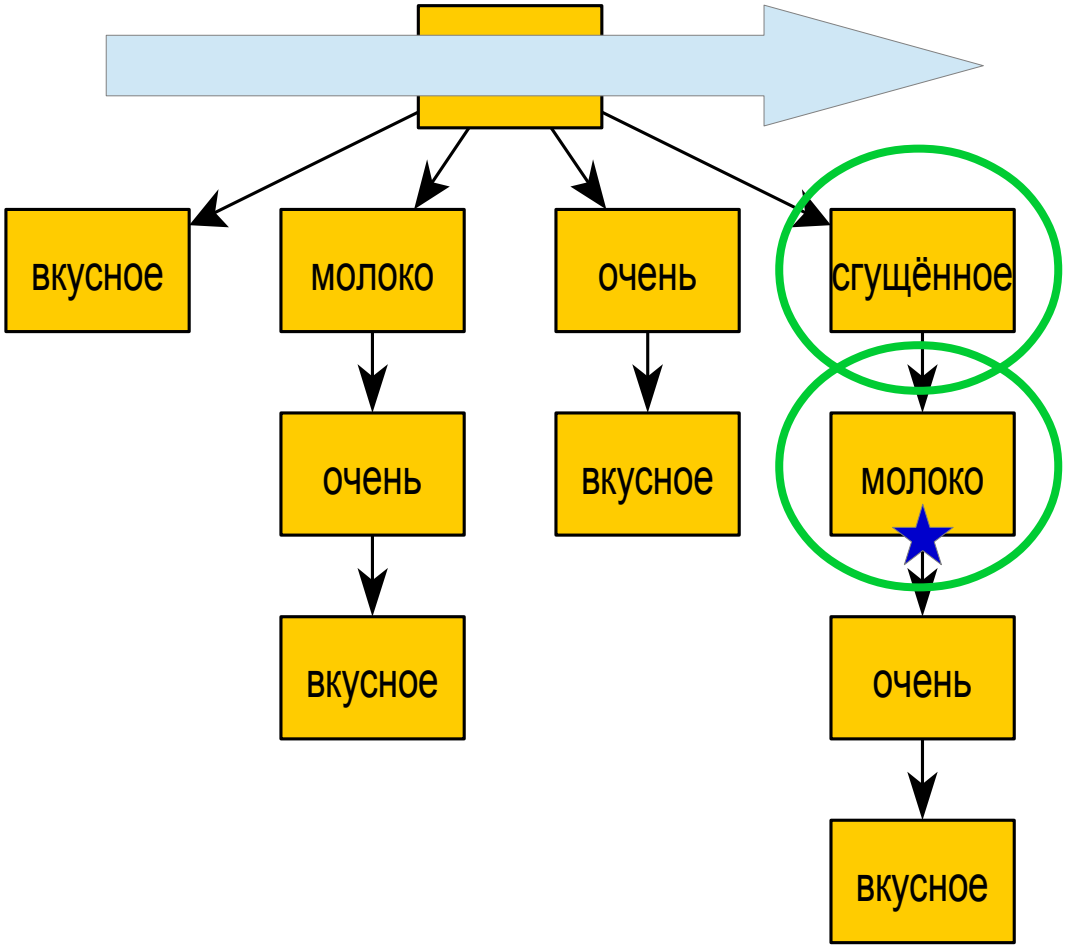
Поиск наибольшей общей подстроки



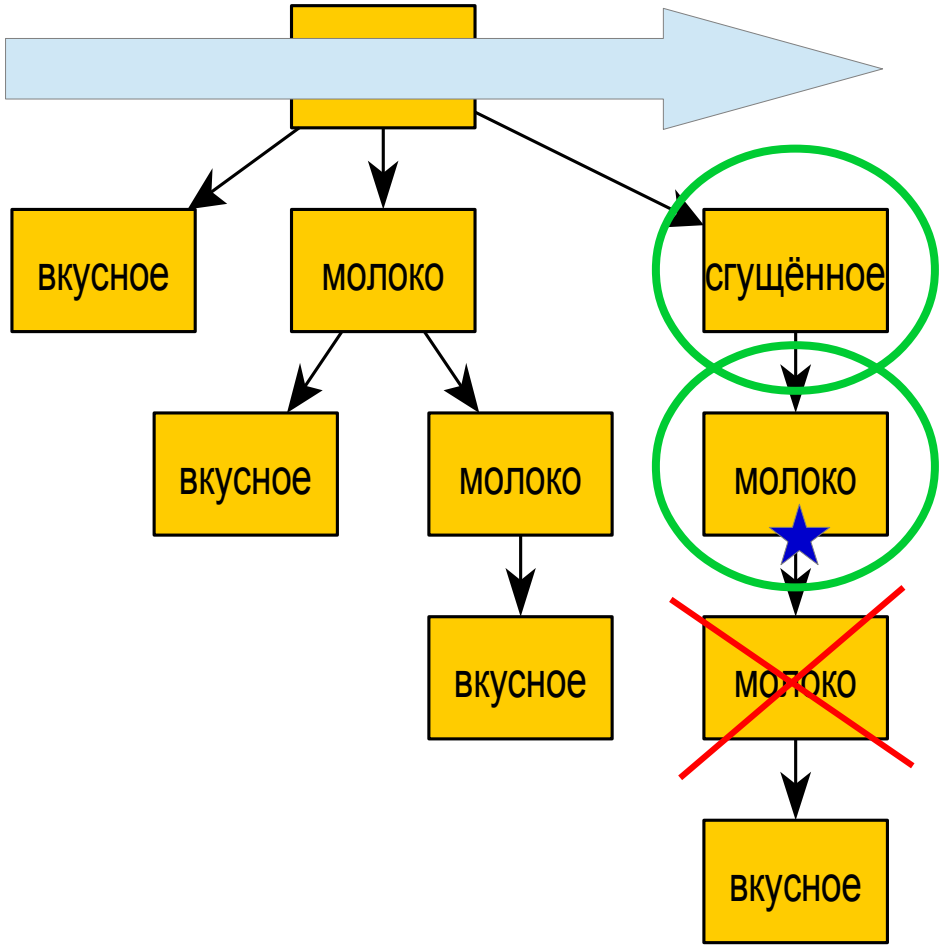


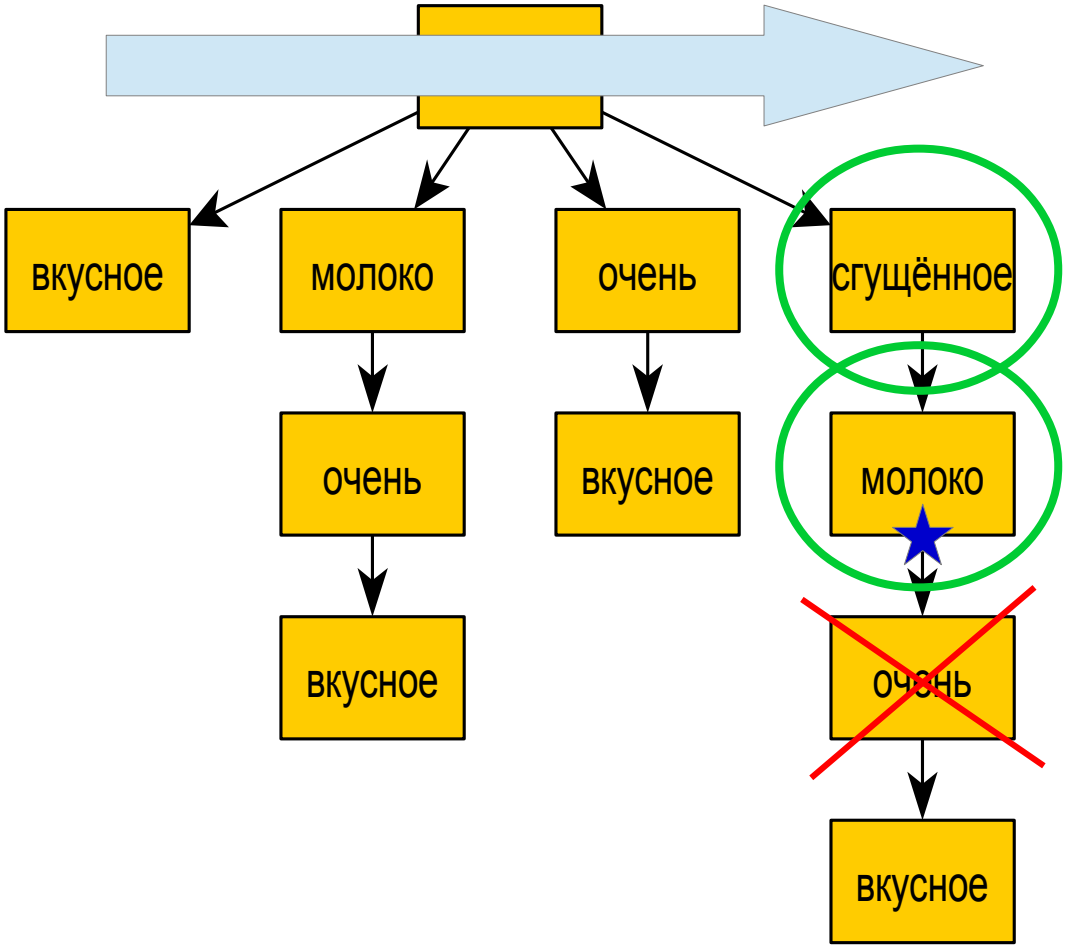
Поиск наибольшей общей подстроки





Поиск наибольшей общей подстроки





Повторять от забора и до...

- Взять первую версию статьи. Установить "метку" автору первой версии на каждое слово.

Повторять от забора и до...

- Взять первую версию статьи. Установить "метку" автору первой версии на каждое слово.
- Взять вторую версию статьи, найти общие строки с первой, вычеркнуть. Для оставшихся установить "метку" автора второй версии.

Повторять от забора и до...

- Взять первую версию статьи. Установить "метку" автору первой версии на каждое слово.
- Взять вторую версию статьи, найти общие строки с первой, вычеркнуть. Для оставшихся установить "метку" автора второй версии.
- Взять третью версию статьи...
- Взять четвёртую версию статьи...

Повторять от забора и до...

- После сравнения последней версии с предыдущей:
  - посчитать авторство текста по количеству меток на словах

Результат

- Мы смогли найти наибольшую подстроку в тексте

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии
- Мы смогли "начислить" авторам баллы

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии
- Мы смогли "начислить" авторам баллы
- Мы смогли построить рейтинг авторов русскоязычного раздела Википедии

Результат

- Мы смогли найти наибольшую подстроку в тексте
- Мы смогли определить авторство каждого слова в тексте статей Википедии
- Мы смогли "начислить" авторам баллы
- Мы смогли построить рейтинг авторов русскоязычного раздела Википедии

- и всё это за 2 недели 100% загрузки CPU, 50 Гб трафика и сна под вентилятор

Результат

Jul-Ar	60 177 баллов	1 355 статей
Valdis72	57 013 балла	3 148 статей
Ghirlandajo	50 702 балла	2 485 статей
Horim	39 765 баллов	969 статей
Alrofficial	36 036 баллов	1 357 статей

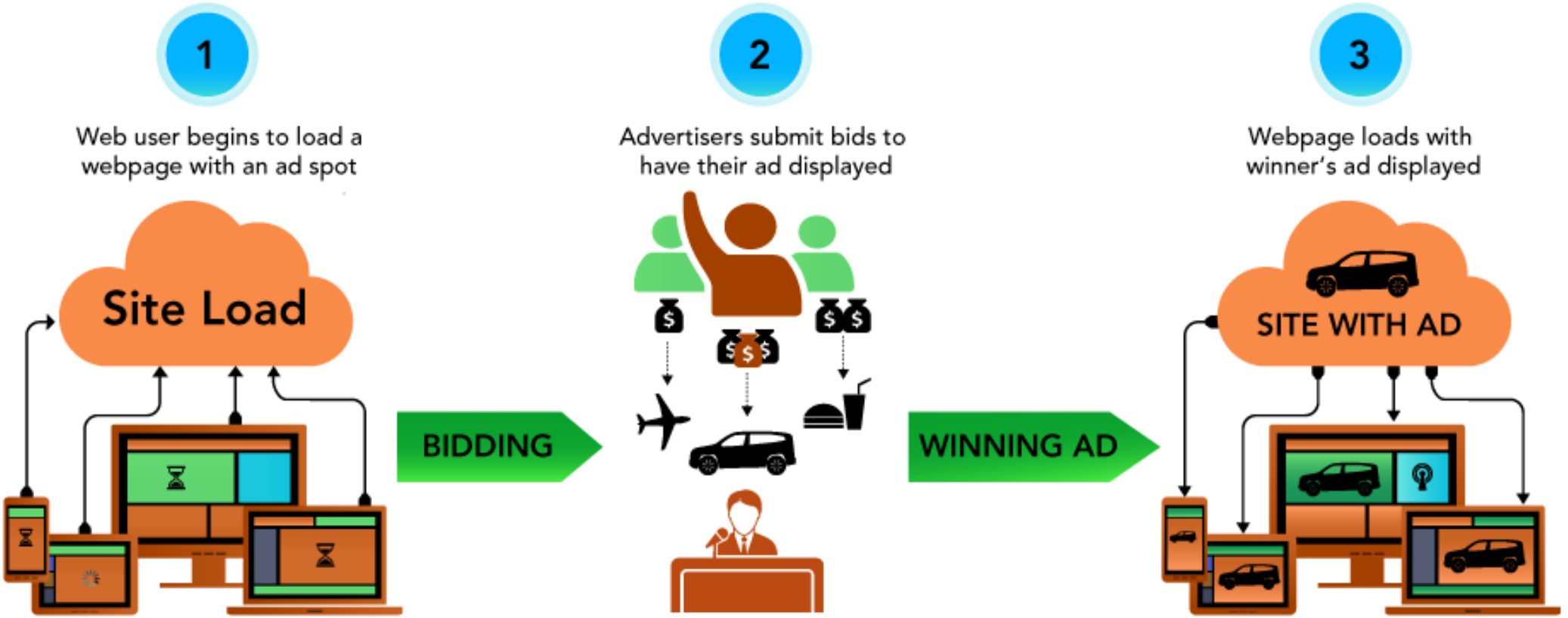
//ru.wikipedia.org/wiki/У:VlsergeyBot/WikiRaiting/2013

Результат

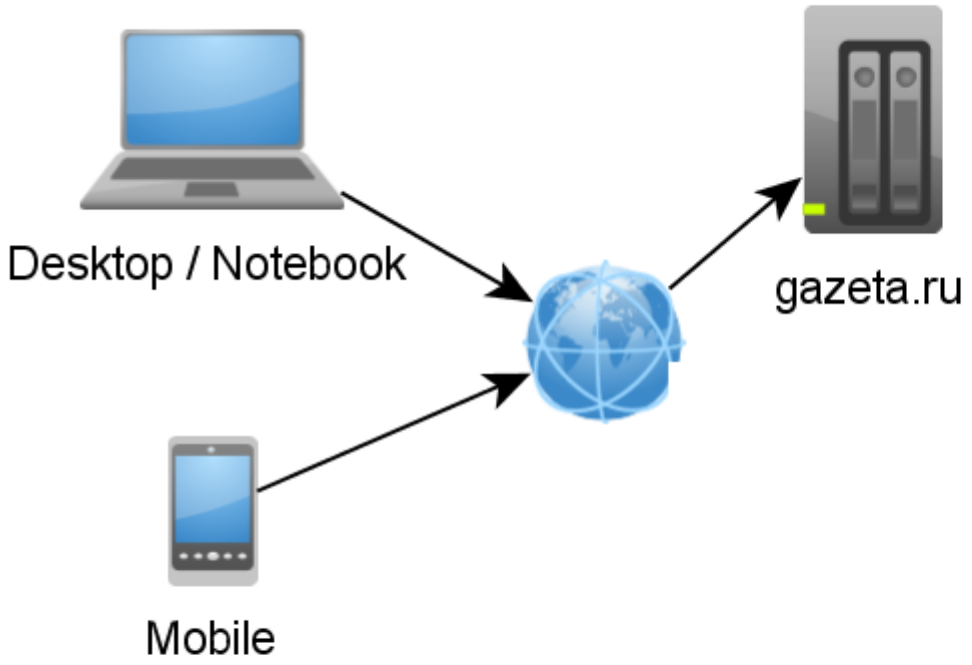
- Люди Икс: Дни минувшего будущего (2 080) — 89,49% (1 861)
- Росомаха: Бессмертный (2 225) — 64,37% (1 432)
- G.I. Joe: Бросок кобры 2 (1 180) — 67,06% (791)
- 300 спартанцев: Расцвет империи (788) — 96,82% (763)
- Человек из стали (1 850) — 41,12% (761)

- Оставшиеся статьи (1 350):
  - Средний вклад: 9,76% ± 21,21%
  - Посещений: 593 ± 736
  - Баллов за статью: 40 ± 93
  - Итого за оставшиеся: 53 704

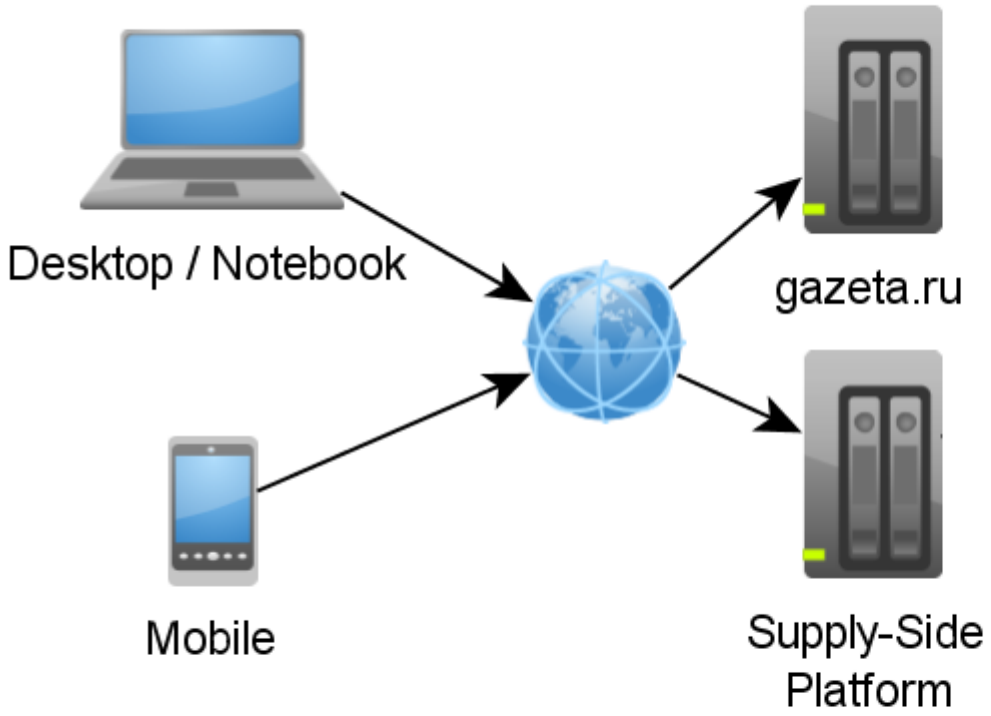
RTB — real time bidding



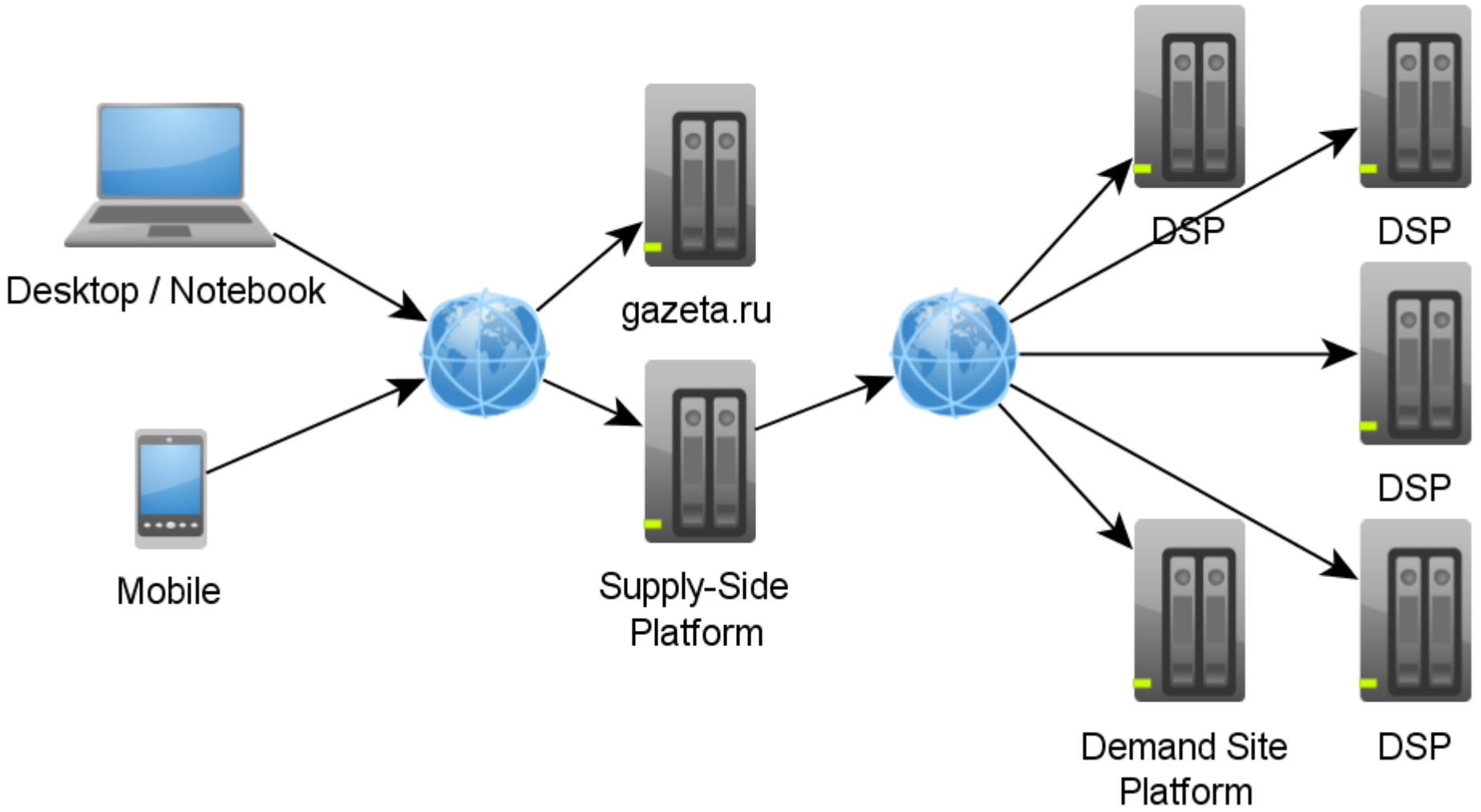
RTB



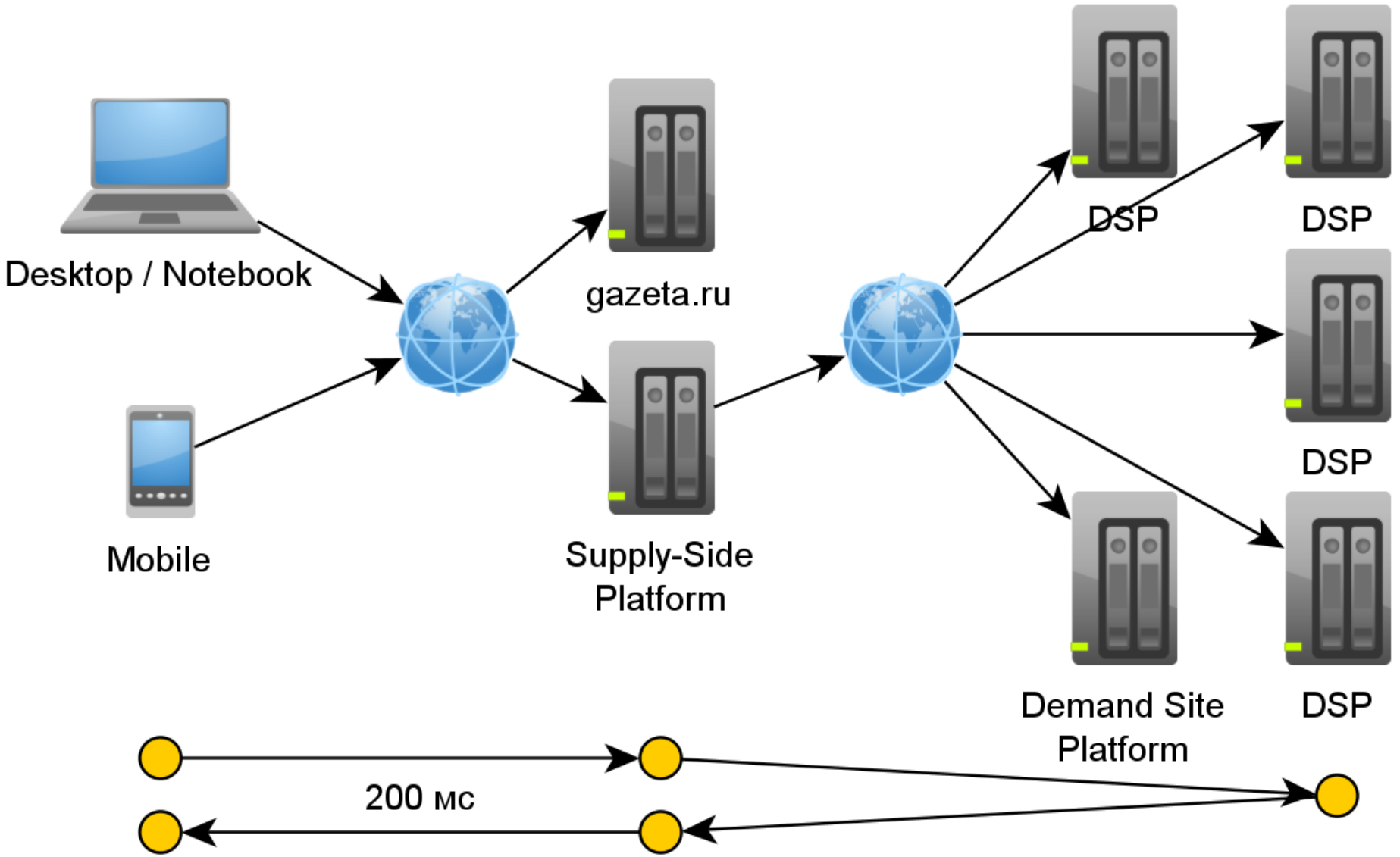
RTB



RTB


Desktop / Notebook
Mobile
gazeta.ru
Supply-Side Platform
DSP
DSP
DSP
Demand Site Platform
DSP

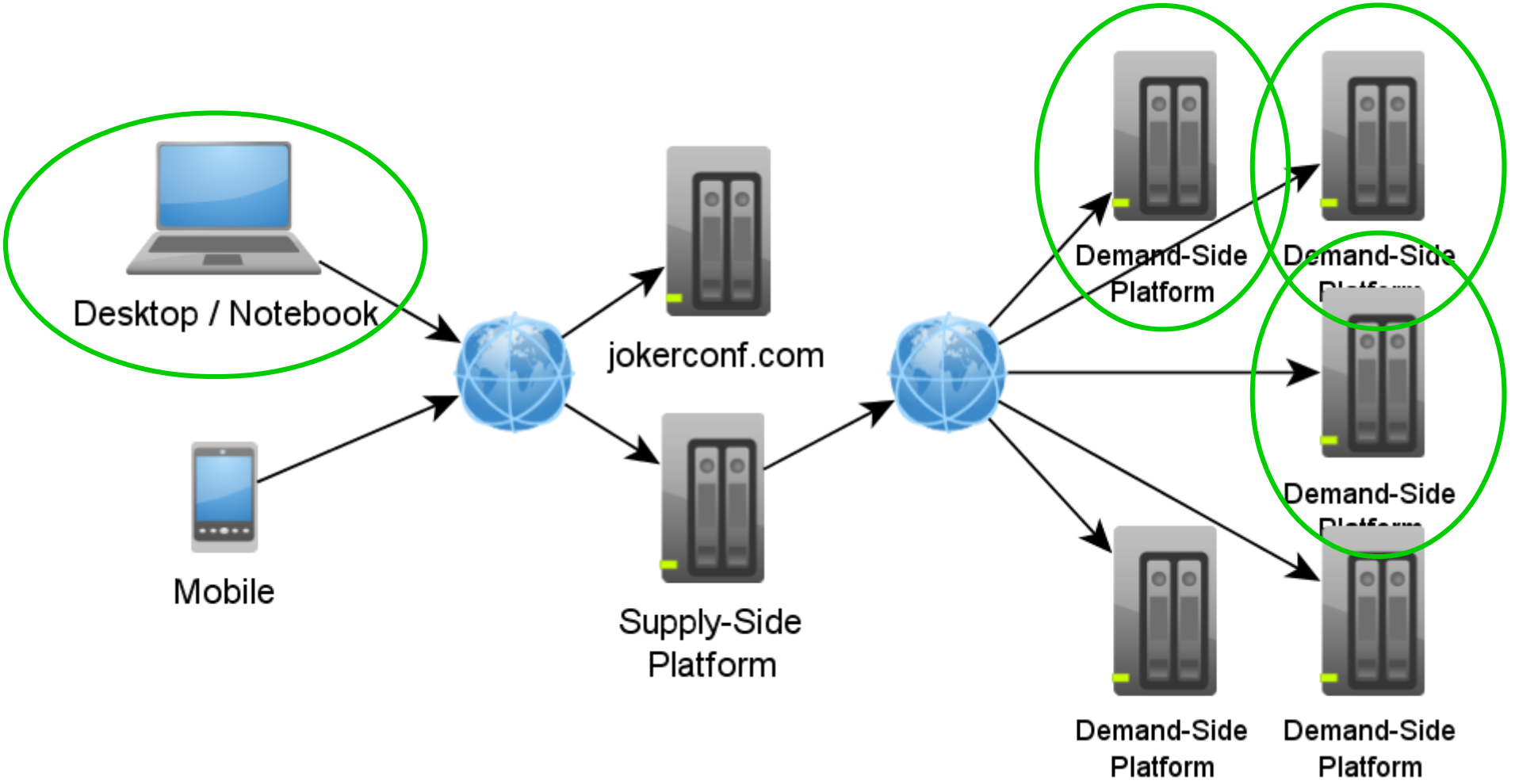
RTB


Desktop / Notebook
Mobile
gazeta.ru
Supply-Side Platform
DSP
DSP
DSP
DSP
Demand Site Platform
200 мс

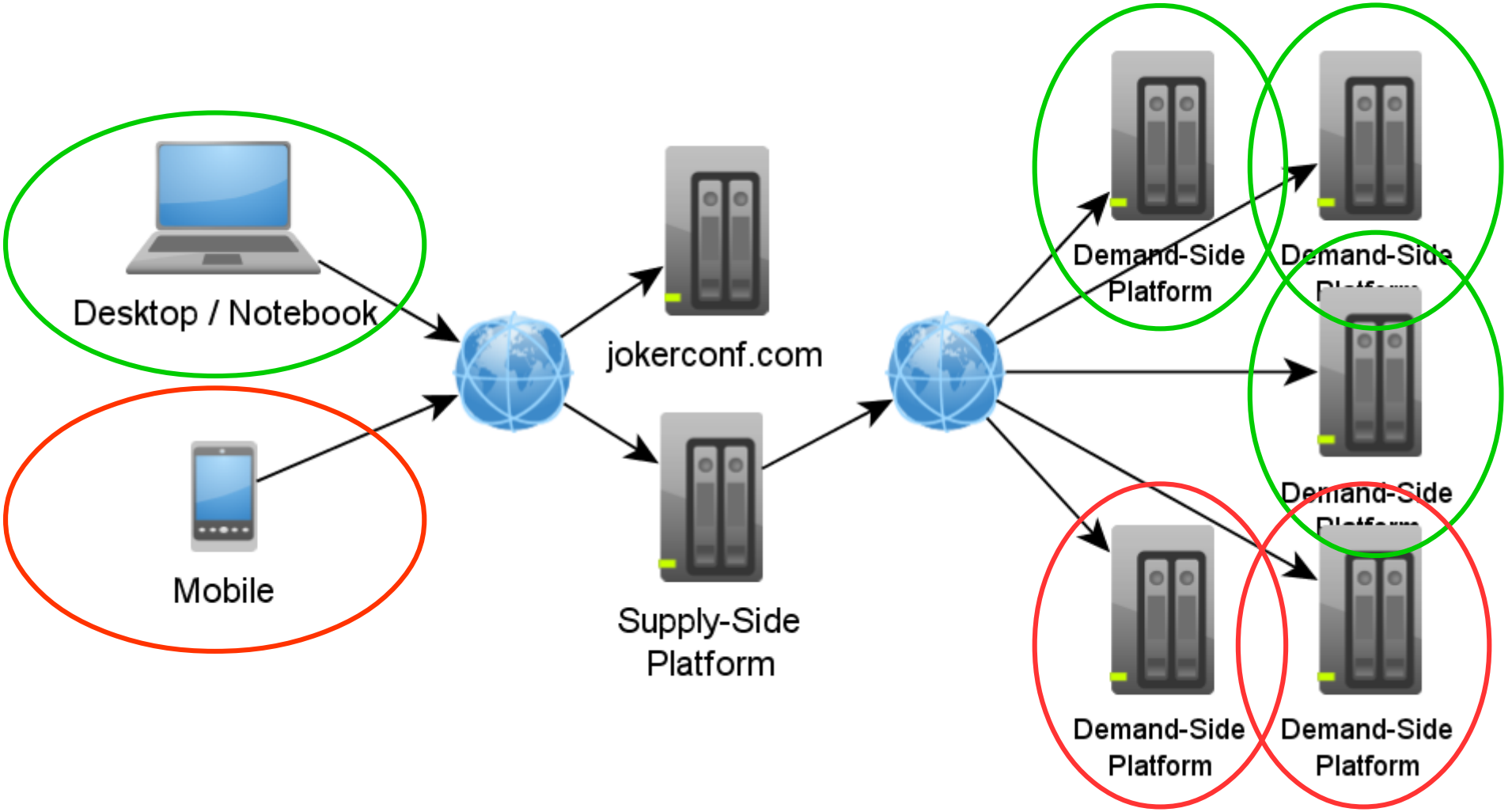
Типичный "SSP"-сервер

- 10 тыс. обращений в секунду
- отношение "входящего" трафика к исходящему от 1 к 5 до 1 к 25
- это когда stop-the-world = потерянные деньги

Предварительная фильтрация



Предварительная фильтрация



Предварительная фильтрация

- Отделить ~~зёрна от плевел~~ настольные браузеры от мобильных

Предварительная фильтрация

- Отделить ~~зёрна от плевел~~ настольные браузеры от мобильных

- Проект «Browser Capabilities Project»
  https://browscap.org/

Предварительная фильтрация

[360Spider]
Parent=DefaultProperties
Comment=360Spider
Browser=360Spider
Browser_Maker=Qihoo 360 Technology Co. Ltd.
Crawler=true

[Mozilla/4.0 (compatible; MSIE 8.0*; *Windows NT 5.1*Trident/4.0*)* 360Spider]
Parent=360Spider
Platform=WinXP
Device_Type=Desktop
Device_Pointing_Method=mouse

Предварительная фильтрация

- Строка UserAgent:
  - `Mozilla/5.0 (Windows NT 6.3; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0`

- Шаблон:
  - `Mozilla/5.0 (*Windows NT 6.3*WOW64*) Gecko* Firefox/37.0*`

Предварительная фильтрация

- Проблема... 287261 шаблон

Pattern Matching

```
for (String pattern : db.keySet()) {
    lines.put(pattern, toPattern(pattern));
}
```

```
Matcher matcher = PATTERN_EMPTY.matcher(useragent);
for (final Map.Entry<String, Pattern> entry :
                                        lines.entrySet()) {
    matcher.usePattern(entry.getValue());
    if (matcher.matches()) {
        return entry.getKey();
    }
}
return "*";
```

Как оценить скорость?

- взять и запустить?

Как оценить скорость?

- взять и запустить?
  - а как же JIT?

Как оценить скорость?

- взять и запустить?
  - а как же JIT?
  - а как же GC?

Как оценить скорость?

- взять и запустить?
  - а как же JIT?
  - а как же GC?
  - а как же (любая другая технология внутри JVM, из-за которой может поменяться скорость выполнения кода)

Используем JMH

- JMH is a Java harness for building, running, and analysing nano/micro/milli/macro benchmarks written in Java and other languages targetting the JVM.
  - http://openjdk.java.net/projects/code-tools/jmh/

Используем JMH

```
@State(Scope.Benchmark)
public class Benchmark {

  private BrowsersCapParser1 parserRegexp;
  private String userAgent = "Mozilla/5.0 (Windows NT 6.1;
WOW64; rv:47.0) Gecko/20100101 Firefox/47.0";

  @Setup
  public void setup() throws Exception {
    parserRegexp = new BrowsersCapParser1();
    parserRegexp.init();
  }

  @org.openjdk.jmh.annotations.Benchmark
  public String testRegexp() throws Exception {
    return parserRegexp.getPattern(userAgent);
  }

}
```

Pattern Matching

```
Result "testRegexp":
  4,404 ±(99.9%) 0,047 ops/s [Average]
  (min, avg, max) = (3,869, 4,404, 4,951), stdev = 0,198
  CI (99.9%): [4,357, 4,451] (assumes normal distribution)
```

- 4,4 операций в секунду ~ 227 мс на "операцию"

Pattern Matching

- 227 мс на поиск шаблона...
- Кто виноват и что делать?

Другое регулярное выражение?

- Вместо 287261 шаблонов использовать один, но большой:

(^*pattern1*$)|(^*pattern2*$)|...|(^.*$)

- Поиск *группы* в шаблоне:

```
final Matcher matcher = pattern.matcher(userAgent);
if (matcher.matches())
  for (int i = 0; i < matcher.groupCount(); i++)
    if (matcher.group(i) != null)
      return this.allPatterns.get(i);
return "*";
```

Другое регулярное выражение?

```
Result "testBigPattern":
  5,939 ±(99.9%) 0,280 ops/s [Average]
  (min, avg, max) = (5,621, 5,939, 6,515), stdev = 0,262
  CI (99.9%): [5,659, 6,219] (assumes normal distribution)
```

- 5,9 операций в секунду ~ 168 мс на "операцию"
- на 35% быстрее!

Другую библиотеку?

- TCL/HSRE RegEx in Java
- https://github.com/basis-technology-corp/tcl-regex-java

- "in drop" замена за исключением экранирования символов: «\\» вместо «\»

Другую библиотеку?

java.lang.StackOverflowError

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

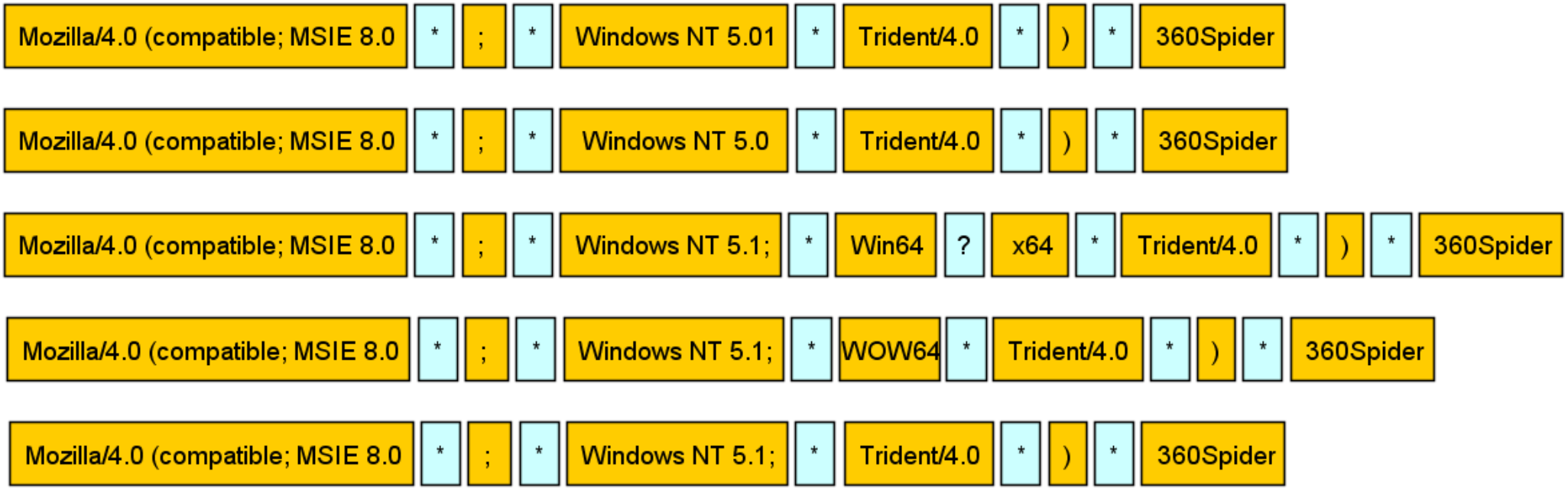
at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

at com.basistech.tclre.Nfa.duptraverse(Nfa.java:340)

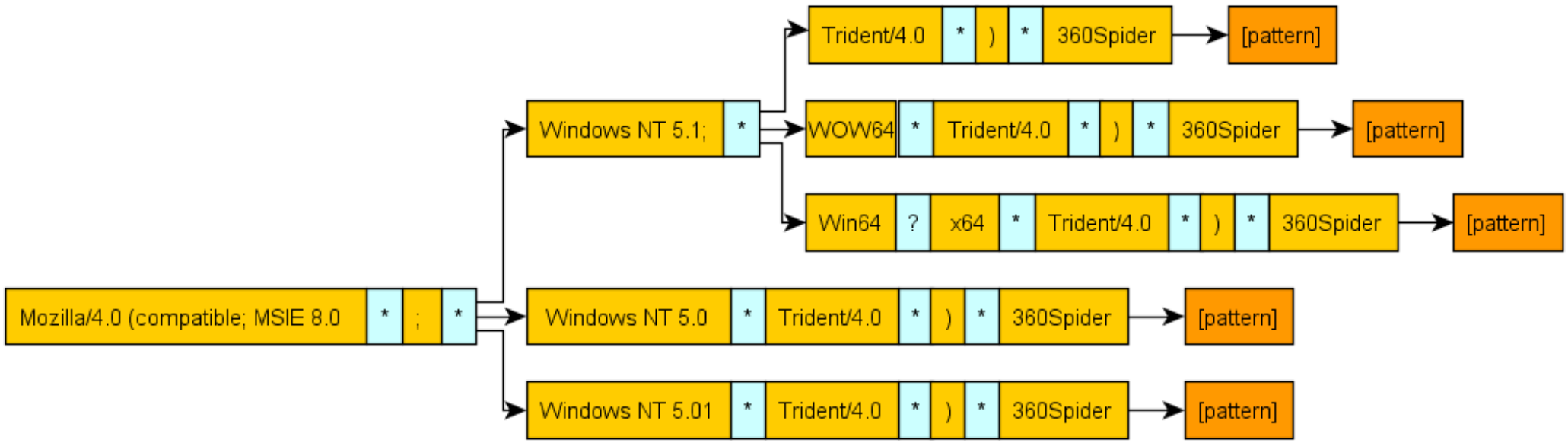
Pattern Matching

- ~~как-то по хитрому использовать regular expression?~~
- ~~взять другую библиотеку?~~
- написать самому?

Pattern Matching



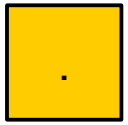
Pattern Matching



Trie

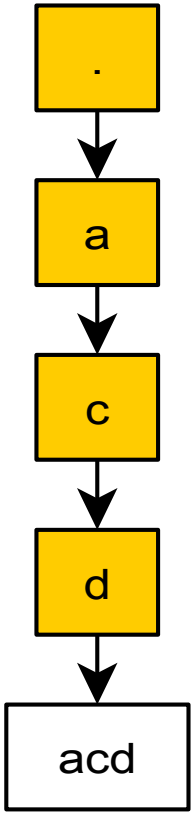
```
class Trie {
    final Node root = new Node('');
}

class Node {

    final char c;
    Map<Character, Node> children;

    String leaf;
}
```

Trie: populate

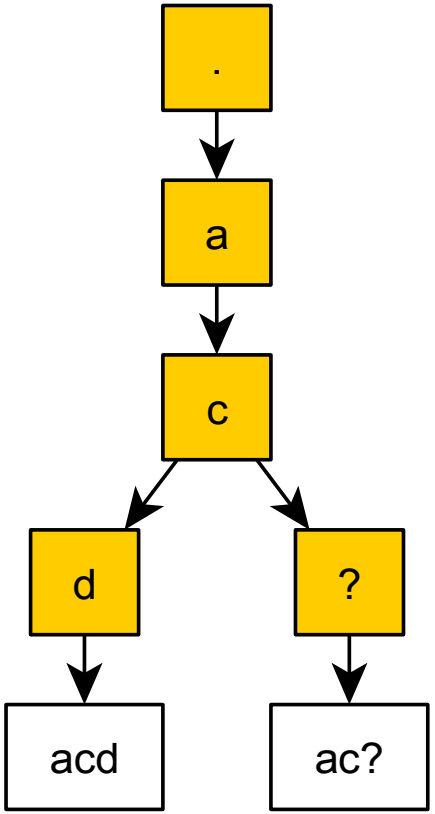


Trie: populate

- "acd"

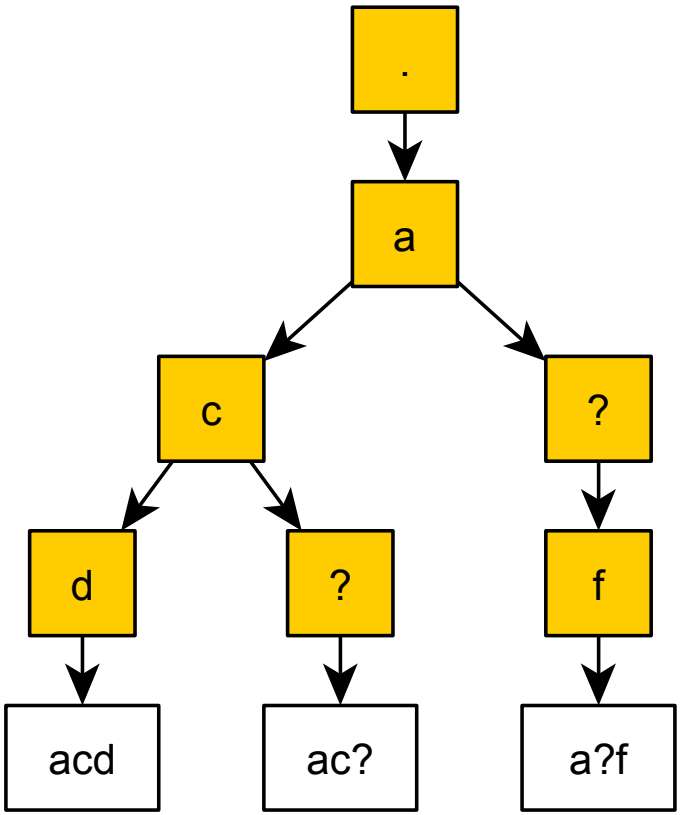


Trie: populate



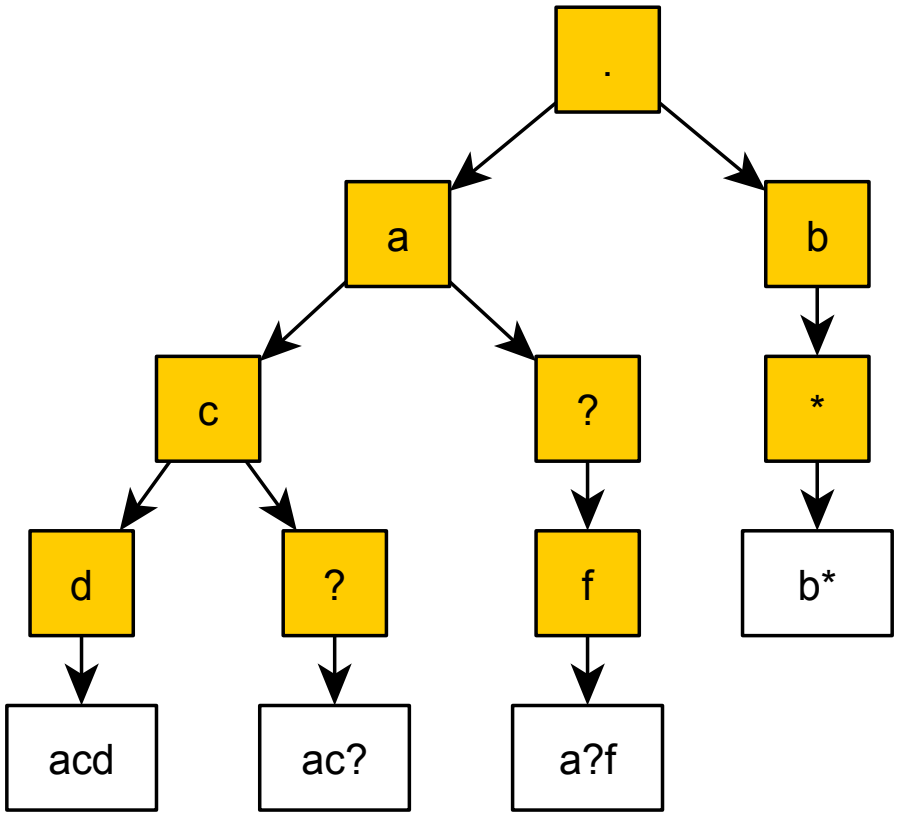
- "acd"
- "ac?"

Trie: populate



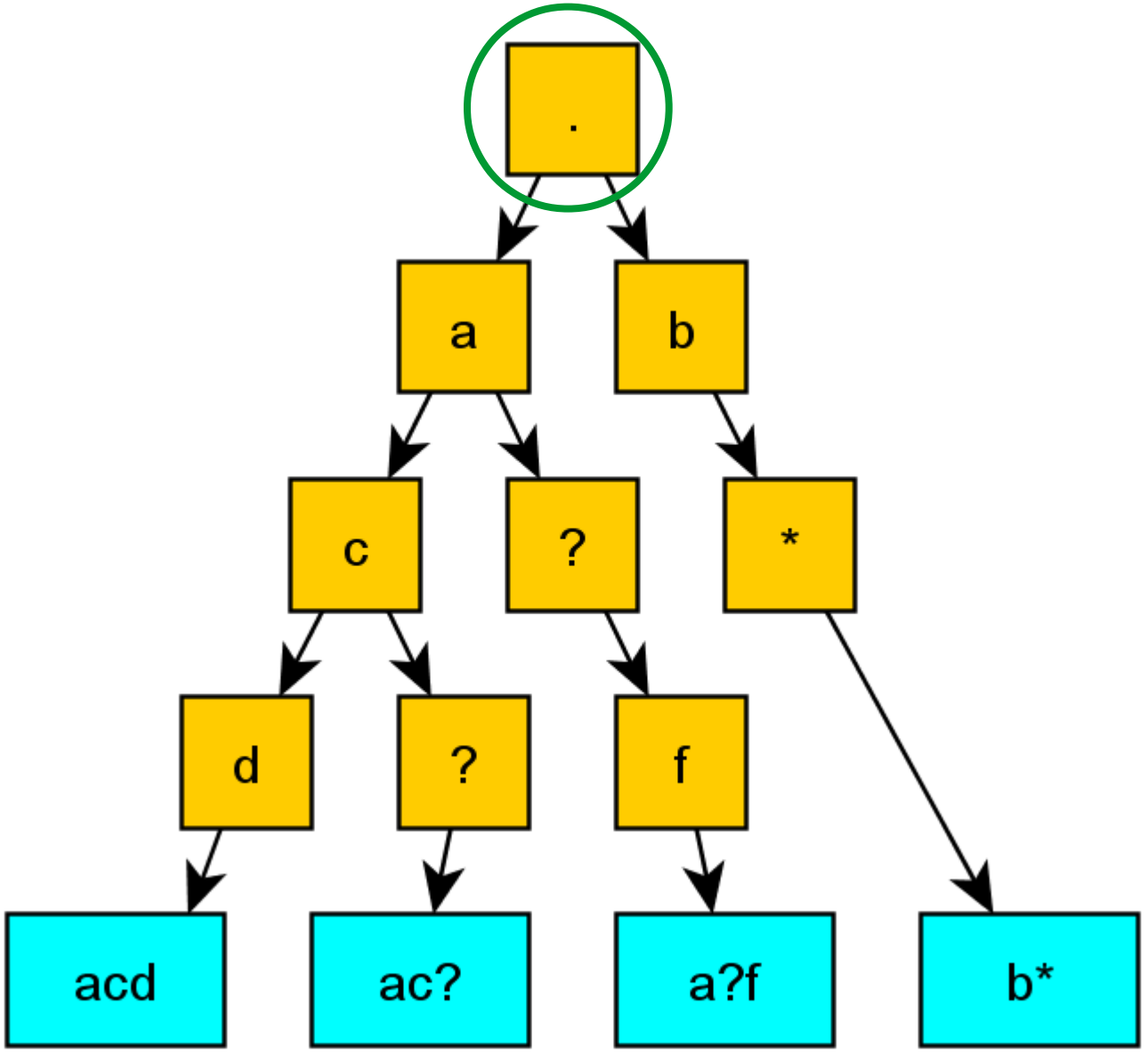
- "acd"
- "ac?"
- "a?f"

Trie: populate



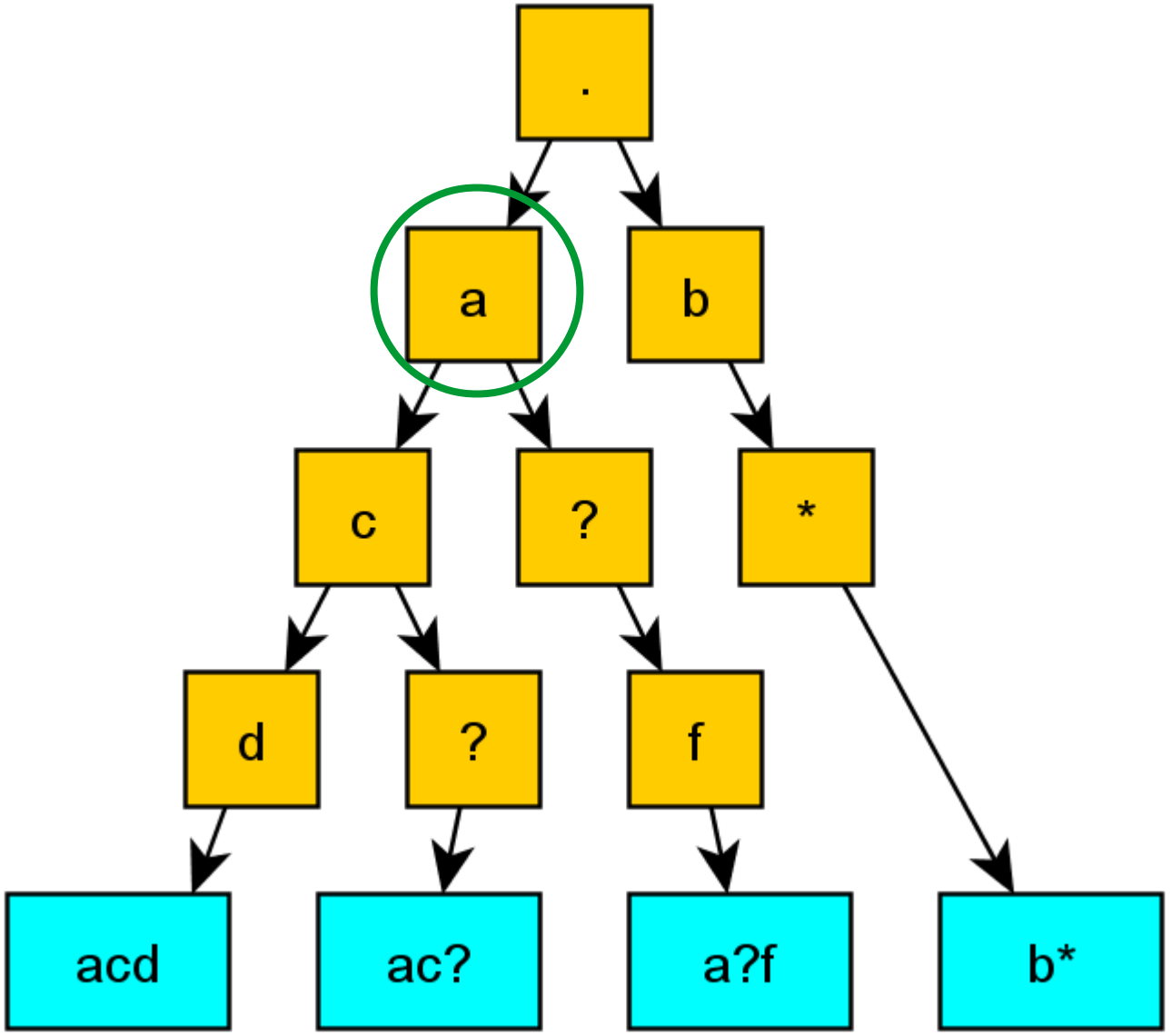
- "acd"
- "ac?"
- "a?f"
- "b*"

Trie: search



вход: "acf"

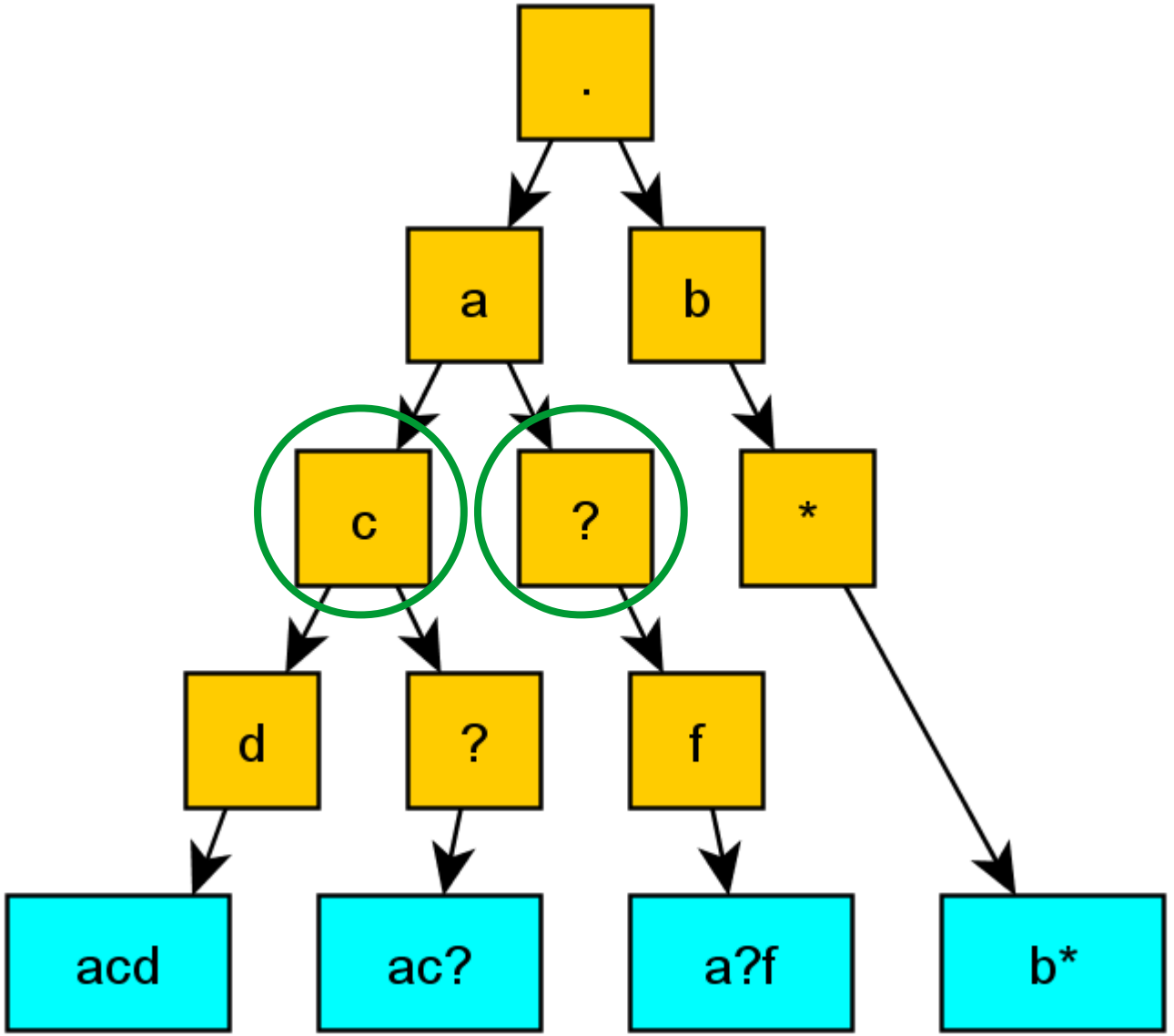
Trie: search



вход: "acf"

1. "a"

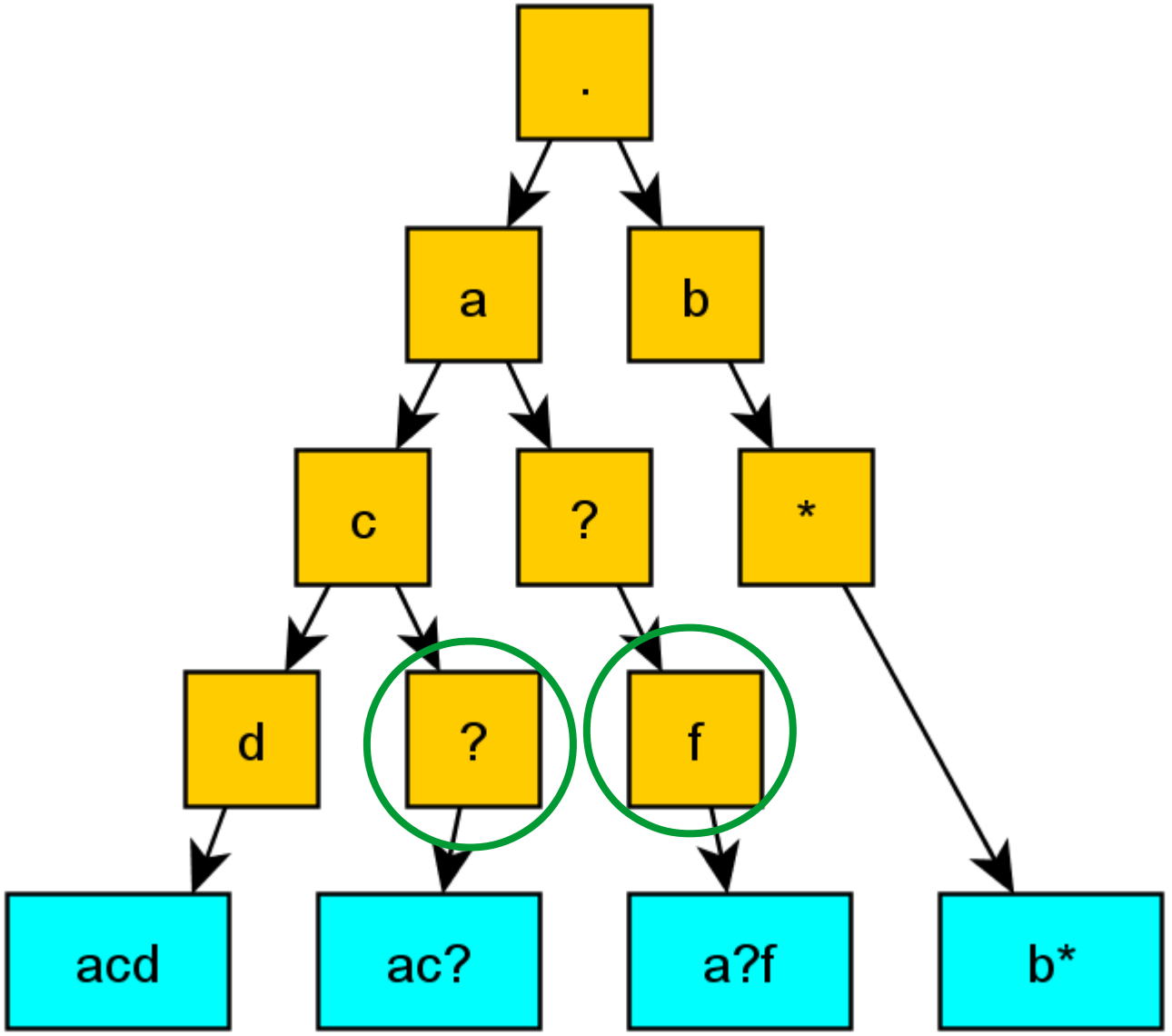
Trie: search



вход: "acf"

1. "a"
2. "c"

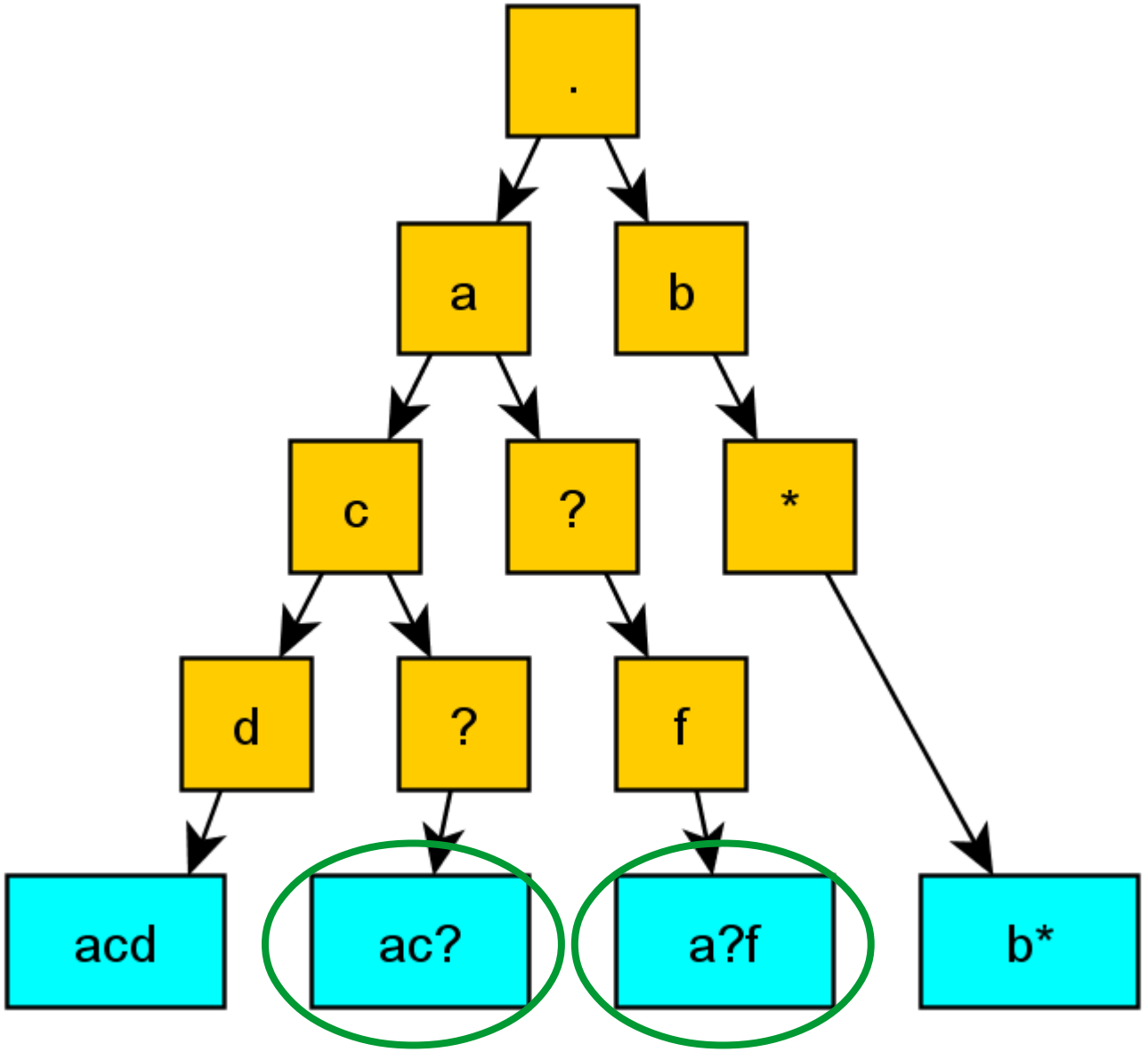
Trie: search



вход: "acf"

1. "a"
2. "c"
3. "f"

Trie: search



вход: "acf"

1. "a"
2. "c"
3. "f"
4. <EoS>

Pattern Matching

- RegEx: 4,404 ± 0,047 ops/s
  ~ 227 мс / строка
- BigPattern: 5,939 ± 0,280 ops/s
  ~ 168 мс / строка
- Trie: 10055,164 ± 125,745 ops/s
  ~ 99 мкс. / строка
- ускорение в 2283 / 1693 раза!

Исходные коды

- http://github.com/vlsergey/Secretary
- http://github.com/vlsergey/jconf2016

вопросы?